PREFACE

This is the user maunal for ESA86/88 E, an economical version of our advanced and popualar microprocessor trainer ESA 86/88-2. The manual describes the hardware and software componets of ESA/86 E and gives the necessary information for interfacing and expanding the systerm,

This manual describes in detail the features offered by ESA 86/88 E Monitor for stand-alone and Serial Mode of operations, Symbolic one-line Assemble, Desassember and debugging features like Single Stepping and Breakpoints. Onboard peripherals, and interface, features like PPIs, USART, Timer, Parallel Printer interface, LCD Interfacing and ESA EPROM, Programmer interface, etc are also discussed in detail in this manual, This document also contains information about communicating with a, Host computer System/CRT terminal using RS 232C Inteface.

the manual is supplement with appemdices containing of useful information about the System Connectors, ASCII character set, Instruction look-up table for Numeric Data Processor and a summary of serial communication details. Complete Schematics and related drawings are also provided in the Appendices.

Please note that this volume is a user guide for ESA 86/88 E and as such does not deal elaborately with the features of 8086/8088 processor and related peripheral and their programming. Details ragarding these can be obtained from the following INTEL publications.

iAPX 86/186 USER'S MANUAL
8086 FAMILY USER'S MANUAL
MICRO SYSTEMS COMPONENT HANDBOOK VOL I AND II

While every effort has been made to present the information in an accurate and simple fashion, we do welcome suggestion for improving the quality and usefulness of this manual.
Please address all your correspondence to:

ELECTRO SYSTEMS ASSOCIATES PVT LTD
4215, J.K. Complex , First Main Road, Subramanyanagar,
P.O. Box: 2139, Banglore - 560 021, INDIA
Phone : ++91 80 23577924 FAX : ++ 91 80 23475615
E-mail : **info@esaindia.com**
www.esaindia.com

CONTENTS

CHAPTER 1	**INTRODUCTION TO ESA 86/88E**	**Page Nos**
1.1	System Capabilites	1- 1
1.2	Specifications	1- 2
1.3	Scope of Supply	1- 4
CHAPTER 2	**CONFIGURATION AND INSTALLATION**	
2.1	ESA 86/88E Configuration	2- 1
2.2	CPU Selection	2- 1
2.3	Operating Mode	2- 1
2.3.1	Baud Rate Selection	2- 2
2.3.2	Printer Enable/Disable	2- 2
2.4	8087 Installation	2- 2
2.5	Memory Selection	2- 2
2.6	Installation of ESA 86/88 E	2- 3
CHAPTER 3	**COMMUNICATION WITH A HOST COMPUTER SYSTEM**	
3.1	Introduction	3- 1
3.2	Installation	3- 1
3.2.1	Installation of WIN 86E	3- 1
3.2.2	Installation of XT 86E	3- 1
3.3	Returning to DOS	3- 3
3.4	Operation Details	3- 3
3.4.1	Download Operation	3- 3
3.4.2	Upload Operation	3- 4
3.4.3	DOS Commands	3- 5
3.4.4	Status Line	3- 5
3.4.5	Command Recall	3- 5
3.4.6	Communication	3- 6
3.4.7	Help	3- 6
3.5	Inter Extended HEX Format	3- 7
3.6	Using Cross Assembler	3- 9
CHAPTER 4	**ESA 86/88E MONITOR COMMANDS**	
4.1	General Operation	4- 1
4.2	Structure of Monitor Commands	4- 2
4.2.1	Rules of Parameter Commands	4- 2
4.2.2	Use of +/- Operators and Registers, in Parameter Specifications	4- 3
4.2.3	Response to Errors	4- 3
4.3	ESA 86/88E Monitor Commands	4- 3
4.3.1	Summary of ESA 86/88 Monitor Commands	4- 4
4.4	Command Output Control	4- 5
4.5	Commands Description	4- 6
4.5.1	Substitute Memory Commands	4- 6
4.5.2	Display Memory Commands	4- 7

4.5.3	Examine/ Modify Register Commands	4- 9
4.5.4	Move Memory Commands	4- 11
4.5.5	Fill Byte and Fill Word Commands	4- 12
4.5.6	Input Byte and Input Word Commands	4- 13
4.5.7	Output Byte ans Output Word Commands	4- 14
4.5.8	Compare Memory Command	4- 15
4.5.9	Go (Execution) Command	4- 17
4.5.10	Breakpoint Facility	4- 18
4.5.11	Single Step Command	4- 19
4.5.12	Help Command	4- 21
4.6	More Monitor Commands	4- 21
CHAPTER 5	**SYSTEM DESIGN DETAILS AND MEMORY ADDRESSING**	
5.1	CPU	5- 1
5.1.1	CPU Reset	5- 1
5.1.2	CPU Clock	5- 1
5.1.3	CPU Address Bus	5- 1
5.1.4	Corrected BHE*	5- 2
5.1.5	CPU Data Bus	5- 2
5.2	System Timing	5- 2
5.3	Memory Addressing	5- 3
5.3.1	Memory Map	5- 3
5.4	Interrupt System	5- 3
5.5	8087 Numeric Data Processor	5- 4
5.6	Bus Expansion	5- 4
CHAPTER 6	**ONBOAD PERIPHERALS**	
6.1	I/O Addressing and I/O Map	6- 1
6.2	Programmable Peripheral Interfaces	6- 2
6.3	Programmable Interval Timer	6- 2
6.4	Serial Interface Controller (USART)	6- 2
6.5	8042 Universal Peripheral Interface	6- 2
6.6	LCD Interface	6- 3
6.7	Parallel Printer Interface	6- 3
6.7.1	Configuration and Operation	6- 3
6.7.2	Theory of Operation	6- 4
6.7.3	Error of Messages	6- 4
6.7.4	Port Specifications	6- 4
6.7.5	Connector Details	6- 5
6.7.6	Direct Output to Printer	6- 5
6.8	ESA 86/88E Connectors	6- 5
CHAPTER 7	**ONBOAD PERIPHERALS**	
7.1	Introduction	7- 1
7.2	Installation	7- 2
7.3	Operation	7- 2

7.3.1	P (PROGRAM) Command	7- 3
7.3.2	R (READ) Command	7- 6
7.3.3	B (BLANK CHECK) Command	7- 7
7.3.4	V (VERIFY) command	7- 8
7.3.5	E (EXIT) command	7- 9
CHAPTER 8	**ESA 86/88E SYMBOLIC ONE-LINE ASSEMBLER**	
8.1	ESA 86/88E Configuration	8- 1
8.2	Assemble Command (A)	8- 1
8.2.1	Format and Operation	8- 1
8.2.2	Assembler Syntax Description	8- 2
8.3.	Assembler Language Conventions	8- 7
8.3.1	Lable field	8- 7
8.3.2	Mnemonic field	8- 7
8.3.3	Operand field	8- 7
8.3.4	Segment Override option	8- 8
8.3.5	More on usage of Labels	8- 8
8.4	Labels Commands	8- 9
8.5	Assembler Directives (Pseudo Op-codes	8- 10
8.6	Disassembly Command (Z)	8- 13
8.7	Addressing Mode	8- 15
CHAPTER 9	**ESA 86/88E MONITOR ROUTINE**	
9.1	Monitor routines dependent on Operating mode	9- 1
9.2	Monitor routines independent of Operating mode	9- 4
CHAPTER 10	**PROGRAMMING EXAMPLES**	
10.1	Familiarization Examples	10- 2
10.2	Illustration of ESA 86/88E Monitor routines	10- 5
10.3	Programming with Onboard Hardware	10- 7
10.3.1	Use of KBINT Key	10- 7
10.3.2	Programmable interval Timer	10- 8
10.3.3	Onboard Programmable Peripheral Interface	10- 9
10.4	Use of 8087 Co-pricessor	10- 10
10.5	Programming with Interface	10- 12
10.5.1	Parallel printer interfaces	10- 12
10.5.4	Liquid Crystal Display Interface	10- 13

APPENDICES

APPENDIX A	Schematics
APPENDIX B	Connector Details
APPENDIX C	Component Layout
APPENDIX D	ASCII Character Set
APPENDIX E	Serial Communication Cable Details
APPENDIX F	ESA 86/88E Assemble Mnemonics Syntax
APPENDIX G	8087 Instruction Set
APPENDIX H	Product List

CHAPTER 1

INTRODUCTION TO ESA 86/88E

ESA86/88 E, is an economical and powerful genenral -purpose mocrocomputer system the can be operated with 8086 or 8088 CPU tha may be used as an instructional and learning aid and also as adevelopment tool in R&D labs and industries.

8086 and 8088 are third generation CPUs from INTEL that differ primarily in their externa data paths. 8088 users an 8- bit wide data bus while 8086 uses a 16-bit wide data bus. ESA 86/88E can be operated with either CPU and the only possible difference would be in the speed of execution (with 8088 CPU, a small speed degradation occurs, because of the 8-bite wide data bus). In either case, the CPU is operated in maximum mode.

The basic system can be easily expanded through the System Bus connector. Powerful features like monitor resident Symbolic One -line asembler and disassembler simplified the programmer's task of entering Assembly languag programs. On-board provision for 8087 Numeric data Processor make ESA 86/88E useful for number-crunching appliction also. Onboard battery backup provision for RAM is made to retain the user programs in the event of a poerr failure or when the trainer is Programmable Interval Timers, USART (for serial communication) PC keyboard Controller and parellel printer interface. futher, ESA 86/88 E A firmware also supports ESA EPROM Programmer interface.

ESA 86/88 E can be operated on single +5 Volts power supplu in stand-alone mode using LCD and optional PC/AT keyboard or in serial mode with a host computer through its RS - 232C interface. user-friendly Windows and DOS Driver packages supplied with ESA 86/88E provide for a powerful and versatile Assembly level programming / debugging environment.

1.1 SYSTEM CAPABILITES

- Assemble 8086/8088 Instruction Mnimonics using ESA 86/88E Symbolic One-Line Assembler
- Disassembler HEX bytes from memory int 8086/88 CPU instruction using monitor resident Disassembler.
- Perform fast numerical computations using the optional 8087 Numeric Data Processor.
- Execute the user program at full speed or debug the program through Single Step and Breakpoint facilities.
- Examine/Mondify the contents of memory location in byte or word Format.

- Examine/Modify the content of CPU registers.
- Write or read data to or from I/O pirts (byte or word format).

- Operations on block of memory such as filling a block of memory with a constant byte or word data, comparing a block of memory with another block, copying a block of data or program within the memory and displaying memory blocks in byte or word format.

- Communicate with a Host PC serially through RS 232C interface at a baud rate of upto 19200 and develop/debug applicatins using the user-friendly Windows or DOS driver packages.

- Supports for downloading user programs into ESA 86/88E from a host computer system in Intel HEX as well as Intel Extended HEX format.

- Support for uploading user programs to Host Computer system and saving them as HEX files on a system.

- Read, Programs, Verify and Blank check of popular EPROMs (2716 through 27512) using optional EPROM Programmer interface module.

- Use the monitor resident Centronics compatible Parallel printer driver softeware and obtain hard copies of serial mode operatoins.

1.2 SPECIFICATIONS

➢ **Central Processor**

8086 or 8088 CPU operating a 5MHz in maximum mode.

➢ **Co-Processor**

On-board 8087 Nummeric Data processor (optional)

➢ **Memory**

ESA 86/88E provides a total of 128 K Bytes of onboard memory.

- 64 K Bytes of ROM using two 27256 EPROMs
- 64 K Bytes of ROM using two 62256 Static RAMs.

Memory addressing and mapping details are given in Chapter 5.

➢ **Onboard Peripherals & intefacing Options**

- **8251A** - Universal Synchronus/Asynchronous Reciver/Transmitter supporting standard baud rates from 110 to 19,200. Baud rate is selected through on-board DIPswitch setting.

- **8253-5-** Programmable Interval Timer; Timer 0 is used for Baud clock generation Timer 1 and Timer 2 are available to the user.
- **8255A-** 3 Programmable Peripheral Interface Provide upto 72 Programmable I/O line. One 8255 is used for controlling LCD and reading DIP Switch. Two 8255s are for the user, of which one is populated by default and the other is optional.
- **8288-** Bus Controller used for generating control in Maximum Mode Operation.
- **8042/8742 UPI** (Universal Peripheral Interface).

➢ **Interrupts**

External:

- **NMI :** 8086/8088 Type 2 Interrupt connected to KBINT key on the trainer. The vectoring information for this interrupt is fully user-defined.
- **INTR :** Available to user on system expansion connector J6.

Internal : Interrupt Vectors 1 (Single step Interrupt) and 3 (Breakpoint Interrupt) reserved for monitor.

➢ **External Interface Signals**

CPU Bus : De- multiplexed and fully buffered, TTL compartible, Addresss, Data & Control signal are available on two 26- pin ribbon cable connectors.

Parallel I/O : 48 Programmable parallel I/O lines (TTL Compatible) through two 26-Pin ribbon cable connectors. Note that only one 8255 and its corresponding 26-pin ribbon cable connector is available as default factory installation, which way additionally be used as a parallel printer interface. Further, ESA 86/88E firm ware uses this 8255 for operations with ESA ESPROM Programmer inteface also.

Serial I/O : RS 232C through on-board 9 pin D-Type female Connector.

PC Keyboard : PS/2 connector is provided for interfacing optional PC Keyboard.

20x4 LCD : 15 Pin flow strip for interfacing the 20x4 LCD.

Timer Signals : Timer 1 and Timer 2 signals are brought to a header.

Power Supply : +5v @ 1A (approx)

Battery backup : 3.6V Ni- Cd Battery as Power Backup to RAM (optional)

All connector details are given in appendix B.

1.3 SCOPE OF SUPPLY (Default)

- ESA 86/88E Trainer Kit with 20x4 LCD
- ESA 86/88E User manual.
- 8086/88 Assembly Reference card.
- RS 232C Cable.
- ESA 86/88 Driver CD-ROM.

CHAPTER 2

CONFIGURATION AND INSTALLATION

This chapter describes all the configuration options and the installation procedures of ESA 86/88 E. Microprocessor traniner. The trainer can be configured in different ways, as determined by DIP Switch Settings. Refer to the conponent layout diagram in appendix C to locate the DIP Switch and the jumpers.

2.1 ESA 86/88E CONFIGURATION

The following table summarizes the DIP Switch settings for different configurations of ESA 86/88E Trainer.

DIP Switch No	*Functionality*	*Configuration*	*Remarks*
1 - 3	Baud Rate Setting for RS 232 C Communication.	Please refer Section 2.3.1	ESA 86/88 E communicates at a max. Baud rate of 19200 with the Windows Driver package.
4	Operational Mode	**ON** : Serial Mode **OFF** : Stand- lone Mode*	
5	Printer Driver Enable	**ON :** Enabled **OFF :** Disabled*	For Serial mode only (refer chapter 6)
6	-	-	Reserved
7	CPU Selection	**ON :** 8086* **OFF :** 8088	Both CPUs Operate at 5MHz in maximum mode.
8	-	-	Reserved

2.2 CPU SELECTION

ESA 86/88E can be operated either with an 8086 or an 8088 CPU operating at a frequency of 5MHz. Install the desired CPU (8086 or 8088) in the socket labeled U22 and select SW7 of the DIP Switch accordingly.

2.2 OPERATING MODE

In stand- alone mode the trainer will be interfaced with the optional PC keyboard. In the seial mode, the trainer can be connected to a CRT terminal or to a host computer system through RS 232C communication standard interface.

2.3.1 Baud rate selection

ESA 86/88E uses 8251 A USART for serial communcation (RS- 232-C). In serial mode of operation. the following initialization are made by system firmware.

- Asynchronous mode
- 8-bit character length
- 2 Stop bits
- No parity
- Baud rate factor of 16 x

Timer 0 of onboard 8253 prvovides the Transmit and Receive Baud clocks for the 8251 A USART. The system firware initializes the timer for proper Baud clock based on DIP Switch settings as described below.

DIP SW1	DIP SW2	DIP SW3	Baud Rate
ON	ON	ON	110
ON	ON	OFF	300
ON	OFF	ON	600
ON	OFF	OFF	1200
OFF	ON	ON	2400
OFF	ON	OFF	4800
OFF	OFF	ON	9600*
OFF	OFF	OFF	19200

(* Factory installed option)

2.3.2 Printer Enable/Disable

ESA 86/88E firmware includes the driver program for Centronics compatible Parallel Printer interface. This driver can be enabled/disabled by setting SW5 of the DIP Switch. Chapter 6 describes installation and operation of this interface in detail.

The printer driver is enabled only in the serial mode and the 26-pin ribbon cable connector J4 is used as the parallel printer interface. Appropriate conversion cables may be optionally obtained from the manufacturer.

2.4 8087 INSTALLATION

ESA 86/88E has on-board provision for 8087 Numeric Data Processor . To install it, just populate the socket labeled U21 with 8087. No other hardware changes are necessary.

2.5 MEMORY SELECTION

ESA 86/88E has four 28-Pin Sockets labeled as U17, U18, U25 and U26 for memory. Of these, Sockets U18 and U26 are populated with 32 K Bytes RAM ICs (62556) providing for a total of 64 K Bytes of RAM. Sockets U17 and U25 are populated with EPROMs containing the firmware. These scokets can be configured for 27256 or 27512 EPROMs by setting jumper JP2.

Jumper Position on JP2	EPROM selected
AB	27512
BC	27256

Refer Chapter 5 for a detailed discussion on the memory mappings.

2.6 INSTALLATION OF ESA 86/88E

To install and operate ESA 86/8 E trainer, the following accessories are required

- **Power supply** : +5V @ 1A (max)
- **For Serial Mode of Operation** : CRT terminal/PC Compatible Host Computer system with RS 232 C interface, driver software and RS-232-C compatible Serial Cables.
- **For Stand-alone operation** : PS/2 compatible PC Keyboard.

The following steps are common for either mode of operation with ESA 86/88E Trainer

1. Select the CPU (8086 or 8088) and Mode of Operation using the DIP Switch.
2. Select EPROM configuration if necessary (Ref. Section 2.5)
3. Install 8087 NDP if desired (Ref Section 2.4)
4. Connect the Power Supply of required capacity to ESA 86/88E (Ref Section 2.6)

Serial Mode Operation :

1. Select mode of operation, Baud rate and Enable/Disable Printer Driver by setting the DIP Switch.
2. Connect ESA 86/88 E to Host PC/CRT through an RS-232-C cable. Interfacing requirements for RS-232-C communication are described in Appendix E.
3. The terminal/computer system should be powered ON and the driver software should be running Refer Chapter 3 for details of installation of Driver Software and Communication procedures

Now the following sign-on message should appear on the console depending on the CPU installed.

ESA 86E MONITOR Vx.y or ESA 88E MONITOR Vxy
(V x.y indicates Version x and Revision y)

The sign-on message is followed by the command prompt, "." in the next line.
The following message appears on the LCD display

ESA 86 MONITOR Vx.y
SER : 9600 P : 86

The message on the LCD gives information about the CPU being used, the mode of operation, current Baud rate configuration (in case of Serial Mode) and the status of the Printe driver.

If the trainer is configured to work with 8088 CPU in Serial Mode with a Baud rate of 4800 and if the printer driver is enabled, the message on the LCD upon RESET will be

ESA 86E MONITOR Vx-y

SER: 4800 P: 88 PRN

Now ESA 86/88E is ready for operation in Serial mode.

No respons in serial mode :

If there is no response from ESA 86/88E in serial mide, after installation as described above check the following.

- Power supply connection as required.
- Configuration of ESA 86/88E (DIP Switch settings, jumpers, etc).
- RS 232C cable connections at both the ends and proper connection of all signals of RS 232C interface (Ref. Appendix E)
- Baud rate configure on trainer should match with that of the serial terminal connected to it Ensure this by setting DIP Switch positions appropriately. Also check for othe serial parameters (Character length ,stop bits, etc)
- If a computer system is the controlling device, check whether the driver program is running and the serial cable is connected to a correct and working serial port.
- Check the message on LCD display, modify the serial parameters as required and verify them by pressing RESET key every time a change is made.

NOTE : DIP Switch staus is read only at Power ON/Reset. If you change the settings, RESET the trainer. If the problem still persists, please contact the manufacturer.

Stand -alone mode opertion:

1. Configure the trainer using DIP Switch and jumper as described in the earlier sections.
2. Install the 20 x 4 LCD and connect a PC Keyboard to the PS/2 or DIN connector provided.
3. Connect the power supply of required capacity to ESA 86/8E and Power ON the trainer.

Now if 8086 is installed, the following sign-on message will appear on the LCD. The sign-on message is followed by the command prompt, "." and th cursor "_" in the next line. In case of 8088 CPU the sign on message will change accordingly.

ESA 86 MONITOR Vx. y
KBD P: 86
._

Now ESA 86/88E is ready for operation in the stand-alone mode.

No response in stand alone mode:

If the trainer does not respond satisfactorily in stand-alone mode, check the following.

- If the LCD backlight is off, check the power supply connections.
- If the display is blank, or shows random patterns check the configuration settings.
- If after a normal sign-on message, there is no response to PC keyboard entries, check for proper connectin at the keyboard connector.

NOTE : DIP Switch is read only at Power ON/Reset. If you change the settings, RESET the trainer. If the problem still persists, please contact the manufacturer.

CHAPTER 3

COMMUNCATION WITH A HOST COMPUTER SYSTEM

3.1 INTRODUCTION

ESA 86/88E operating in the serial mode, can be connected to CRT terminal or a host computer system. when a computer system is the controlling element, it must be executing driver software to communicate with ESA 86/88E.

ESA 86/88E is supplied with Windows and DOS driver packages, which allow the user to establish communication between the trainer in serial mode and a Host PC through its asynchronous communication ports (COM 1 or COM 2). Both the packages fully support the ESA 86/88E Serial Monitor commands including file upload and download facilities. HEX files generated by PC native and Cross-tools can also be downloaded to the trainer using these packages. Further the data from ESA 86/88E memory can also be uploaded to the disk file o f the computer. The compatiable serial cable for RS-232-C communication is also included in the package.

3.2 INSTALLATION

a) Configure ESA 86/88E for serial mode of operation and set the serial port of ESA 86/88E for 9600 Baud and No parity (Refer Chapter 2).

b) Connect the compute system to ESA 86/88E over the COM 1/ COM 2 Serial port using the serial cable provided. (Refer to your system's Technical Manual for details regarding the definitions using the signal definitions on COM ports. The signal definitions of RS-232-C port of ESA 86/88A can be found in Appendix E.

3.2.1 INSTALLATION OF WIN 86E

Win 86E is a Windows based communiction package for ESA 86/88E trainer that provides a powerful and convenient debugging and deveopment environment to the user. The user must install this software from the ESA 86/88E Drivers CD.

Insert the CD in the available drive and run Setup. EXE from the ESA 86/Win 86Es folder. the Setup program will guide the user through the rest of the installation procedure. once the software is installed successfully, Win 86E offers a complete online help for working with the commands. Under win86E offers a complete online help for working with the commands. Under Win 86E the traniner can communicate at a baud rate of up to 19200.

3.2.2 INSTALLATION OF XT86E

XT86E is the DOS based communication software supplied as a .EXE from the ESA

86/88 Drover CD-ROM and can be executed on a PC compatible computer system under PC-DOS/MS-DOS operating system.
Insert the CD in the available drive and run **XT86E.EXE** from the **ESA 86E\XT86E** folder. Alternatively, this file may be directly copied to and executed from the Hard disk also.

Now the following message appears on the screen.

XT86E Version x.y

ELECTOR SYSTEMS ASSOCIATES PVT LTD.

BANGALORE

Alt + S	**-**	**Set Communication Parameters**
Ctrl + F1	**-**	**Help**
Alt + F1	**-**	**Command Help**
<Esc>	**-**	**Clear command**
<F1>	**-**	**Previous Command Character**
<F3>	**-**	**Command Recall**
Ctrl + U	**-**	**Upload Command**
Ctrl + D	**-**	**Download Command**
!Command	**-**	**DOS Shell/ Command**
Alt + X	**-**	**Exit**

Press any key to Continue

XT86E Checks for the presence of communication ports COM 1 & COM 2 . If both port are not available it will display the message

No serial port present as reported by BIOS

and exit to DOS. Otherwise XT86E will read the communication Parameters from the file " XT86E. INS" and initialize the communication port. XT86E searches the current directory for the file "XT86E.INS". If the search fails it will search the path given by the DOS environment variable INIT. If the file is not present, the following message is displayed.

XT86.INS NOT FOUND!
SERIAL PARAMETERS ARE SET TO COM 1, 9600, 8,2 NONE
DO YOU WANT TO CHANGE?

YES **NO**

If option "No" is selected the communication parameters: serial port COM 1 Baud 9600, Data bits 8, Stop bits 2, and parity none are set. If option "yes" is selected the Communication Parameters can be interactively modified as described in section 3.4.6

Now XT86E attempts to establish communication between the computer and ESA86/88E, If successful. the command prompt "." appears on the screen. subsequently during the Power-ON or RESET the following sign- message appears on the screen followed by the command prompt.

ESA 86E MONITOR Vx.y or ESA 88E MONITOR Vx-y

If communication fails then the console will display

Unable to transmit data

Retry Ignore Serial Pars

If ESA 86/88E is not powered on, power it on and press<R> to retry establishing the communiction. check the configuration and setting of the trainer and retry. pressing <I> will exit XT86E to DOS. pressing <S> will allow the user to change the serial parameters of the Host system, if communication is still not established, check for the following .

- Ensure that ESA 86/88E is conneted to the correct COM PORT and that the COM port is functiining properly
- Ensure the ESA 86/88E is functioning properly and configured correctly for operation in Serial Mode.
- Check the RS 232C cable connnections.
- Since the communication package utilizes the hardware handshake signal DTR while communicating with ESA 86/88E, the interfacing cable must support this signal also.

Note: XT86E utilizes an interrupt driven routine for reading characters from the COM port. thus it is possible for XT86E to miss some characters if the system has any resident programs that are interrupt driven. (For example, many systems include a CLOCK program in the AUTOEXEC file, to display the time on the upper right corner of the screen,) Hence it is desirable not to run any such resident programs while XT68E is running.

If the problem persists, Please contact the manufacture.

3.3 RETURNING TO DOS

Use can terminate XT86E and return control to DOS by typing **<Alt+X>** when the program is waiting for keyboard input.

3.4 OPERATIONAL DETAILS

the complete command set of ESA 86/88E is transparent and is fully supported by XT86E. Chapter 4 and Chapter 7 of this manual contain a detailed discussion of these commands. Press **<Ctrl + F1>** for on-line help. In addition , XT86E supports the file dowmload/upload and other commands , which are explained below.

3.4.1 DOWNLOAD OPERATION

This feature allows downloading of the contents of an object code file into ESA 86/88E memory. To perform download operation, type **<Ctrl + D>** in response to the command prompt **(".")**.

The system will now prompt for the name of the disk file, from which the information is to be downloaded as follows:

Download filename [.HEX]:

Enter the file name with extension , terminated by <CR>. if the filename is invalid, it displays "file not found !" and prompt again for the filename. If the path specified is invalid, it displays an message " path not found " and prompts again for the filename if none of the above errors occur , the system will read the file gather the data in the specified address range, reformat the data for compatibility with the protocol required by ESA 86/88E and send the data to the trainer. During this proccess, the system will display

Downloading in Progress: XXXX

where XXXX is the start-address of the record being downloaded. After downloading, the object the system returns to the command prompt of ESA 86/88E

Note: The object code file must be a "HEX" file with records in INTEL Extended HEX format. Please refer the relevant INTEL manuals for the defination of INTEL Extended (16-Bit) HEX format. Most of the cross assemblers for 8086/8088 produce object code files which are "HEX" files with records in Intel Extended HEX format.

3.4.2 UPLOAD OPERATION :

This feature allows the user to upload data from the memory of ESA 86/88E to the computer system and save it in the specified disk file in INTEL 8-Bit HEX format.

To perform upload operation, type **<Ctrl +U>** in response to the command prompt (".."). the system will Prompt for the file name, Segment Address, Offset start and end addresses as follows.

Upload filename [HEX] :

Segment address :

Start address :

End address :

upload filename is the name of the disk file, into which the information is to be uploaded. Enter the file name the file name with extension, terminated by <CR>. If the file already exists. then the system will display

File already exists!
Overwrite?

Yes No Append

Select the first option by pressing <Y> to overwrite the contents of the existing file. Pressing <N> will let the user specify another file name. Select th third option <A> to append to the contents of the existing file. If no error occurs, the system proceeds by prompting the Segment address.

The user has to enter segment from which the program or data has to upload. After this the system will prompt for starting address of the program. Enter the Starting address

terminated by **<CR>**. A maximum of four hex digits is allowed for the starting address.next, the system will prompt for the ending address of program or data from ESA86/88E memory. Enter the ending address terminated by **<CR>**. A maximum of four hex digits is allowed for the ending address.

Then the system will gathe the data in the specified address range of the ESA 86/88E ; reformat the data intp INTEL 8 - Bit HEX records and store the data in the specified file. While the uploading is in progress, the system will display the starting address (XXXX) of each record being uploaded.

Uploading in progres : XXXX

Once the uploading is complete XT86E will let the user specify another address range. User may specify a new address range or enter <Esc> to terminate uploading operations.

The following error messages may appera during upload and download operatins.

1 Invalid function number!
This is XT86E internal error: therefore contract ESA 86/88E techincal support for assistance.
2 File not found !
3 Path not found !
4 No more files!
5 Access denied !
6 Insufficient disk space !
7 Invalid file handle !
8 Unable to continue upload !
9 Colon is not present at the start of the Record !
10 Invalid data in (source file) the start of the Record !
11 Checksum Error in the following Record !

Please check for validity of file records in the event of any of the above errors.

3.4.3 DOS COMMANDS

At the command prompt "." any vaild DOS command can be entered preceded by "!". XT86E environment is saved and the dos command is executed. then XT86E environment is restored and XT86E command prompt displayed again.

3.4.4 STATUS LINE

During the session, some commonly used XT86E commands are displayed at the bottom line in reverse video for the convenience of user. the status line is displayed as **Ctrl + F1 help, Alt+F1 Cmd Help, Alt + S Commn, <Esc> Clr Cmd, Alt +X Exit, F1, F3,** V,v.

3.4.5 COMMAND RECALL :

The feature facilitates recall o f the commands already entered by the user, upto a maximum of 16 commands. Press <F3> to recall the previous command. All the

commands are kept in a circular buffer.User may use up-arrow and down-arrow keys to traverse through the sequence of commands in backward or forwaed direction respectivly in a circular fashion, user may recall just the previous command , character by character, by pressing <F1> desired number of times. using <Esc> key any time before pressing **<CR>** can clear current command being entered.

3.4.6 COMMUNICATION

Communication parameters can be set during the session by pressing **<Alt +S>**. List of parameters and their current values will appear on the screen. Select the desired parameter with the help of arrow keys and keep the space bar **<SP>** pressed till the desired value appears.

The Paramenters that can be set are Communication Port (COM 1 / COM 2). Baud Rate (110/150/ 300/600/1200/2400/4800/9600), Parity bits (7/8), Number of Stop bits (1/2), and parity (NONE /ODD/ EVEN). After selecting the desired values press **<CR>** to set the parameters or press **<Esc>** to ignore the values.

Communication parameters can also be modified, while user is in DOS by editing the file XT86E. INS. This file contains single data line, having five integers separated by blanks, representing various communication parameters. These five integers represent serial communication port, baud rate, number of data bits, number of stop bits and parity, in sequence.

The following shows details of the integer values and correspinding parameters.

1		2		3		4		5	
Communication Port	Intg	Baud Rate	Intg	Data Bits	Intg	Stop Bits	Intg	Parity	Intg
COM1	0	110	0	7	0	1	0	ODD	0
COM2	1	150	1	8	1	2	1	None	1
		300	2					Even	2
		600	3						
		1200	4						
		2400	5						
		4800	6						
		9600	7						

NOTE : The baud rate listed above are those supported by the DOS Driver package XT86E. The trainer is capable of communication at baud rates other than these also. It may be noted that ESA 86/88E is capable of communicating at baud rates other than these also. it may be noted that ESA 86/88E is capable of communicating at a maximum baud of 19200. However this baud rate is suppported by the windows communication package Win86E only.

3.4.7 HELP

On-line help is available for all ESA 86/88E monitor commands and specific commands of XT86E. Help facility can be selected by Crtl + F1. A menu of commands is displayed from which desired command can be selected by using arrow keys and help information about that command is displayed

in the remaining part of the screen. context sensitive help is available using Alt+F1 this facility can be used if more information is desired about the command being entered against command prompt.

3.5 INTER EXTENDED HEX FORMAT

The files uploaded/downloaded under the control of XT86E are in the INTEL. Extended hex format. In this format, binnary values are coded in ASCII. for example, the binary value 0100 1111(= 4FH) is coded as two ASCII characters "4" AND "F". In other words, the byte 4FH is represented by two bytes 341 (4 in ASCII) and 46H (f in ascii). Thus this representation requires by two bytes 34H (4 in ASCII) and 46H (F in ASCII). Thus this repesentation requires twice as many bytes as required by binary representation.

A file in extended hex format can consist of upto four types for records. Each record begins with a record field containing 3AH the ASCII Code for colon (:). This is followed by a RECORD LENGTH filed, which specifies the number of bytes of information that follow in the record is 255. After the data the record has a checksum field that contains the ASCII representation of the two's complement of the eight-bit sum of the bytes that result from converting each pair of ASCII hexadecimal digits to one byte of binary, from and including data field. the RECORD LENGTH field to and including the CHECKSUM field is zero. the record ends with carriage return, line feed sequence (0DH, 0AH).

The four types of records that may be present in a file of extended Hex format are :

- Extended Address Record
- Start Address Record
- Data Record
- End of File Record

Extended Address Record:

RECD MARK	REC LEN	ZERO	REC TYPE	USBA	CHECK SUM
":"	" 02 ''	"0000"	"02'	XXXX	XX

This record is used to specify the bits 19-4 of the Segment Base Address (SBA) where bits 3-0 are zero. Bits 19-4 of the SBA are referred to as the Upper Segment Base Address (USBA). Subsequent data bytes are loaded at the specified offsets Relative to this USBA. The Extended Address Record may appear anywhere within the file. This value remains in effect until another Extended Address Record is encountered.

Example Record:

:02 0000 02 0200 FA

02+0000+02+0200+FA = 00 checksum validity

USBA = 0200H. For example, third byte in a subsequent data record with a load address of 0100H is loaded at 0200;102.

Data Record :

RECE	REC	LOAD	REC	_.DATA._	CHECK
MARK	LEN	ADDRESS	TYPE		SUM
“ : ”	XX	XXXX	“00”		XX

The data record provides a set of ASCII characters that represent the hex digits of the data bytes. The method of loading these data bytes has already been described in the discussion of the extended address Record.

Example Record:

:06 00 10 00 BA FF FF B0 40 90 B2
06+00+10+00+BA+FF+FF+B0+40+90+B2= 00 check sum validity

Load address = 0010 & Number of data bytes =06

Assume the USBA Currently valid is 0200. then the data is to be loaded as shown below:

0200:0010=BA	0200:0011=FF	0200:0012=FF
0200:0013=B0	0200:0014=40	0200:0015=90

Start Address Record:

RECE	REC	LOAD	REC	CS	IP	CHECK
MARK	LEN	ADDRESS	TYPE			SUM
“ : ”	“04”	“0000”	“03”	XXXX	XXXX	XX

The start address record is used to specify the execution start address for the object file. This record can appear anywhere in the .HEX file If the start address is not present, CS and IP values will remain what they were before downloading.

Example Record :

: 04 00 00 03 02 00 01 00 F6
04 +00 +00 +03 +02 +00 +01 +00 +F6 = 0 check sum validity.

After Loading. CS = 0200 and IP 0100

End of File Record :

RECD	REC	ZERO	REC	CHECK
MARK	LEN		TYPE	SUM
“.”	“00”	“0000”	“01”	“FF”

This is a record of fixed structure and is used to specify the end of the .HEX file.

3.6 USING CROSS ASSEMBLER

A convenient way of creating a file to be downloaded into ESA 86/88E is to use a cross -assembler for 8086/8088 that can generate the object code in Extended Hex Format.

Example: chapter 10 includes some programs that have been developed using X8086. The distribution diskette contains HEX files (object code) for these example the code is in extended hex format and the files were created using X8086 and the corresponding linker LINK. The user can download these files using the Download command and execute them according to previously given guidelines.

The steps involed in creating a .HEX file are:

1. Create a source file using the DOS text editor or notepad and save it as *filename ASM*
2. Assemble *filename* ASM using a cross assembler to create filename. OBJ
3. Link the single file filename .OBJ

Specify code offset according to the memory organized in the trainer. You may even skip this prompt if "ORG" directive has been used in the source file. Select E (for Extended Hex) as the option for output file format. This process creates filename .HEX that can then be downloaded to ESA 86/88E.

Downloading .EXE files and .COM files .

- If the source program in assembly language is assembled using the Macro Assembler MASM and then linked using the Microsoft Linker, the resulting object code file will have .EXE file structure.
- .EXE files have certain limitations in their structure and hence can be converted to .COM files, which are convenient for executing small programs on PC/XT/AT Systems under PC DOS/MS DOS.

ESA 86/88E supports downloading of .EXE files with total size less than 64K and .COM files, however after converting them to suitable HEX format. The conversions can be accomplished using the following file converter packages available from ESA Pvt.Ltd.

1. EXE2HEX.COM for converting .EXE files to .HEX files
2. COM2HEX.COM for converting .COM files to .HEX files

Detailed documented information about these converter packages will be supplied with the packages if opted by the user.

CHAPTER 4

ESA 86/88E MONITOR COMMANDS

ESA 86/88E Microprocessor can be operated either in serial or stand-alone mode. This chapter describes the serial and keyboard monitor cammands used for working with the trainer. Most of the monitor commands in either mode of operation bear uniformity in syntax and operation. The following discussion on Monitor command is therefore common to either mode of trainer operation, unless otherwise specified.

4.1 GENERAL OPERATION

Installing ESA 86/88E, selecting the mode of operation and its appropriate configuration is described in chapter 2. It is essential that the trainer should be configured properly for working with it smoothly, Getting a proper sign-on message upon RESET as explained in the previous chapter is the minimum requirement for working with ESA 86/88E

Reset Status: On Power ON/RESET, all information about the previous user program is lost and the registers may acquire new data. However the contents of user RAM are not disturbed if onboard RESET is used. If the RAM is backed up with battery, then the user RAM data is not lost even if power is switched OFF. Resetting the trainer initializes the segment & status registers of the CPU as described below.

Table 4.1 Register Initialization

Resgister	Value (Hex.)
CS (Code segment)	0
DS (Data segment)	0
ES (Extra Segment)	0
SS (Stack segment)	0
IP (Instruction Pointer)	0
FL (Flags)	0
SP (Stack Pointer)	100

Further, interrupt vectors 1, 2 and 3 are initialized as follows:

Interrupt 1: Single step interrupt- This interrupt is used by the monitor for implementing single step command.

Interrupt 2: NMI(non-maskable Interrupt)- This interrup is implemented by the on-board KBINT key and the vectoring information is completely user defined.

Interrupt 3: Breakpoint Interrupt- This interrupt is used with Go command and its user is at the user's discretion.

A detaild disussion of the 8086/8088- interrupt vectors can be found in the Intel's 8086

family user's manual (chapter 2, Processor Control and Monitoring interrupts PP 2.22-2.28)

Whenever the monitor is re-entered as a result of a single step or brealpoint interrupt, the monitor saves the contents of the 8086/8088 registers on the user stack and subsequently restores the register contents from the stack before it prompts for command entry. Since the SP register is intialized to 0100H, and memory loctions 0H-CFH are reserved for monitor, the stack length reserved for the user is 48 bytes (i.e location D0H-FFH).

Of these, 26 byte must be left for register contents, should one of the above interrupts occur (leavin 22 bytes for the user). In addition locations 100H to IFFFH are used for other system functuons and system tables. Hence the user RAM starts form 0:2000H

4.2 STRUCTURE OF MOINTOR COMMANDS

When the monitor is ready to accept a command from the user it output a period ('.') as the command prompt character at the beginning of a new line.

The commands entered by the user consist of one or two-character command mnemonic followed by a list of command parameters. This list may be up to three parameters long depending on the command being used. when more than one parameter is required a single comma ('.') is used between the parameters as a separator. A command is terminated either by carriage return or by escape depending on the command itself. Commands are executed one at time and only one command is allowed in a command line.

4.2.1 RULES OF PARAMETER ENTRY

When a command reuired the entry of parameters from the user, the following rules apply.

All addreses in the 8086/8088 systems consist of a segment value and an offset value. The segment value is enterd first, a colon (' : ') is entered as a seperator and then the offset value is entered. If the segment value is not specified (note that in such a case, the colon also should be omitted), the default segment value is the current content of the code segment (CS) register. The address is entered is a hexadecimal value, most significant character first. the valid range of hexadecimal values for an address entry (either segment or offset) range from 0000 to FFFFH. If more than four digits are entered, only the last four digits entered are valid. In other words all address values are interpreted modulo 64K.

Data is also entered as hexadecimal value, most significant character first. The valid range of data entries is 00 to FFH for byte entries and from 0000 to FFFFH for word entries. If more than two (for byte entries) or four (for word entries) digits are entered, only the last two or four digits entered are valid.

Examine/Modify register command (X) requires register symbols as parameters. The register abbreviation entries required by the X command are described in detail with the explanation of this command.

4.2.2 USE OF +/-OPERATORS AND REGISTER, IN SPECIFICATION PARAMETERS

While the address/data parameters are to be entered as hexadecimal values, the operators '+' and '-' can be used to form expressions and also to specify the use of the contents of any 8086/8088 register as address/data values.

For example, suppose we wish to specify an address whose segment is (ES) + 10H and whose offset is (BX) - 30H. One-way is to calculate these values as shown below,

Assume (ES) = 0270H and (BX) = 0080H
Segment value = (ES)+ 10H = 0280H
Offset value = (BX)- 30H =0050H

Then the address can be specified as 0280:0050

However, ESA 86/88E supports direct specification of such an address value as shown below:
ES + 10: BX -30

The systim will automatically evaluate such expressions and use the correct values. These perators can be used to considerably simplify the specification and checking of relative addresses also.

4.2.3 RESPONSE TO ERRORS

Whenever an error is detected the command is aborted the symbol ('?') is output on the command line, a carriage return and linefeed are issued and the command prompt character ('.') is output at the beginning of the new line.

Command execution occurs only a command terminator (a comma or a carriage return depending on the command) is entered. Hence a command entry can be cancelled any time before the terminator is entered. The monitor detects entry of any character that is not legal or does not match the expected entry. The monitor then aborts the cimmand, displays a '?' and returns to command entry mode.

4.3 ESA 86/88E MONITOR COMMANDS

The Monitor commands in both serial and stand-alone mode are similar and the following discussion holds good for trainer operation in either mode. These commands are summarized in the following table and are described in detail in later sections.

In both the table and individual command descriptions, the following notation is used.

[V]	indicates that 'V' is optional
[V]*	indicates one or more optional occurrences of "V"
<V>	indicates that 'V' is a parameter to be entered by the user.
,/-	indicates that either of the two characters is to be entered.

NOTE: Thes symbols are used to clarifyl the command formats and they are neither to be entered by the user nor output by the system.

In the descripiton of the individual commands. it is assumed that 8086 CPU has been installed.However, all the commands work in identical fashion even if 8088 CPU is installed. The only observable difference, would be in the sign-on messsage generated by the system following Power ON / RESET.

4.3.1 SUMMARY OF ESA 86/88 MONITOR COMMADS

COMMAND	FUNCTION	FORMAT/SYNTAX
S	Substitude memory bytes: Displays/modifies memory bytes	**S[<address>]<CR>[,/][<new data>] ,]* <CR>**
SW	Substitude memory Words: Displays / Modifies memory in byte	**S[<address>]<CR>[,/][<new data>] ,] * <CR>**
D	Display memory bytes: display block of memory in byte format	**D<start address> [<end address>] <CR>**
DW	Display memory Words: displays block of memory in word format	**DW<start address> [<end address>] <CR>**
X	Examine/modify registers: displays modifies 8086/8088 CPU registers	**X[<reg><CR>[,<new data>/,]] <CR>**
M	Move Memory : copies a block of memory from one location to the other.	**M[<Start address>,<end address>, < destination address><CR>**
F	Fill Memory (Bytes): Fills a block of memory with constant byte data	**F<start address>,<end address><byte value> <CR>**
FW	Fill Memory (Words): Fills a block of memory with constant word data	**FW<start address>,<end address><word value> <CR>**
I	Input byte: Accepts and displays the data byte at the input port	**I<port address>.<CR>[,]*<CR>**
IW	Input word : Accepts and displays the data word at the input port	**IW<port address>. <CR>[,]*<CR>**
O	Output byte: Outputs a data bytes to the output port	**O<port address>. <CR>[<data>/,]*<CR>**
OW	Outer Word: Outputs a data word to the output port	**OW<port address>.<CR>[<data>/,]*<CR>**
C	Compare memory: Compares a block of memory with another block	**C<start address1>.<end address 1>,<start address2><CR>**

G	Go (Execute): Transfers the processor control from the monitor to the user program address with optional brealpoint	**G<CR>[< start address>[,<breakpont address>]]<CR>**
N	Single step: Executes single instruction of the user program	**N<CR>[<start address>,],[,/<new address>]*<CR>**
H	Help command: Lists Monitor commands with their valid syntax	**H[<Command mnemonic>] <CR>**
P*	Invoke programmer software: Invokes the software for ESA EPROM Programmer Interface	**P<CR>**
A**	Enter assembler: Invokes esa 86/88E Symbolic on-line Assembler	**A[address]<CR>**
LL**	List lables: List all lables defined in the symbol table	**LL<CR>**
LC**	Label Clear: Clears all previously defined labels from the symbol table	**LC<CR>**
Z**	Disassembly command: Disassemble Hex Code into 8086 mnemonics for a specific memory range	**Z[<start address>[,<end address>]]<CR]**

** Refer chapter 7 for a detailed discussion of this command*
*** Refer chapter 8 for a detailed discussion of these commands*

4.4 COMMAND OUTEPUT CONTROL

1. In Serial Mode:

During serial operation, the output to the console can be stopped using the following method. This is applicable in cases of commands where in the monitor continuously outputs data to the terminal viz. Display memory, Disassembly and compare memory commands.

CTRL+C, entered at any time, immediately terminates the command and the monitor return to the command entry mode. Entering **CTRL+S** stops the output to the terminal but does not terminate the command, Entering **CTRL+Q** now resumes the output from the point at which it has been stopped. Entering **CTRL+C** now terminates the command.

Note that **CTRL+S** should be followed only with **CTRL+Q or CTRL+C.** No other console input is allowed following a **CTRL+S.**

2..In Stand-alone Mode:

when working in stand -alone mode the LCD is refreshed when the display on row 4 is completd. the next display occurs on Row 1 of the LCD. In case of commands where large display is involved, such as 'Display Memory' the LCD displays a total of 16 bytes or 8 words at one time and waits for the user to enter <CR> to proceed with subsequent displays. Also thers is an additional <CR> required in stand-alone mode for the monitor to return to command entry prompt. However such commads can be terminated at any stage by entering<Esc>.

4.5 COMMAND DESCRIPTION

This section describes the command supported by ESA 86/88E monitor. The discussion hold good for either mode of trainer operation unless otherwise mentioned.

4.5.1 SUBSTITUTE MEMORY COMMANDS

FUNCTION

Substitute memory byte (S) and substitute memory word (SW) commands are used to examine the contents of the selected memory locations. If the contents of the memory location can be midified e.g., a RAM Location), the contents can optionally be modified.

SYNTAX

S[<address>]<CR>[,/][<new data>],] * <CR>
SW[<address>]<CR>[,/[<new data>],] *<CR>

OPERATION

1. Both commands operate in a similar fashion. To use either command, enter S or SW when rompted for command entry, then enter the address of the memory location to be examined/ modified. This is followed by carriage return.
2. The monitor will now output the contents of the adressed location followed by a hyphen the monitor's data entery prompt character) and a space to indicate that the addressed locations can now be modified.
3 To modify the contents of this location, enter the new data value. Note that when using the SW command, the byte contents of the next consective memory location (addressed memory location +1) are output first, followed by the byte contents of the actual location addressed. similarly, when updating memory contents using the SW command, the first byte entry will be written into the next consecutive memory location, and the second byte entry will be written into the addresed memory location.
4. After optionally modifying the contents of the addressed loction, type a comma to examine/ modify the next two consecutive memory location or previous two memory locations (SW commad)
5. A <CR> terminates the command.

ERROR CONDITIONS:

1. Trying to modify non-existent or ROM locations.
2. Entering non-Hex parameters as data.

EXAMPLES :

1 Examine RAM location 2100 H, relative to the DS register, modify the ontents of location] 2101H and 2102H and examine the contents of 2101H agian.

.S DS:2100 <CR>
0000:2100 A5-,
0000:2101 FF-B7,
0000:2102 FF-5A
0000:2103 FF-<CR.
.
.S DS:2101 <CR>
0000:2101 B7- <CR>

Note that here DS is set at 0000. Hence the segment value displayed will be 0000.

2. Examine ROM location FFOO:9CH and trying to modify the same.

.S FFOO : 9C<CR>
FFOO : 009C FF - 44, ?<CR>

Note that in ROM location data can not be modified. Then it will give error.

3. Examine word at location 2120 H relative to DS register. Assume that DS is set to 2000

.SW DS:2120, <CR>
2000:2120 A1F4 <CR>

(Note that here A 1H is at location DS:2121 H while the byte at the addressed location DS:2120 is F4H)

4.5.2 DISPLAY MEMORY COMMANDS

FUNCTION

Display memory byte (D) and Display memory word (DW) comands enable the user to output the contents of a block of memory, either in byte or word format.

SYNTAX

D <start address>[,<end address>]<CR>
DW <start address>[,<end address>] <CR>

OPERTATION

1. Both the commands operate in a similiar fashion.when operating in serial mode, these commands provide a line-formatted output of the memory block starting at the'start address' and ending at the 'end address'. However in case of stand-alone operation a total of 16 bytes or 8 words is displayed at one time and the monitor wait for a <CR> to refresh the output and display the next 16 bytes or 8 words of data if required.
   Note that the 'end addres' is always relative to the segment value specified with 'start address', or implied with 'start address' (i.e., the contents of CS register if no segments value is specified). Thus with each command execution, a maximum of 64K bytes or 32K words can be displayed.

   To use either command enter D (for byte output) or DW (for word output) when prompted or command entry. Then type the start address of the memory block. Now, if only byte or word is to be displayed enter the carriage return. On the other hand, if a block of memory is to be displayed enter the end address followed by a carrige return.

2. The monitor will now output, beginning on the next line, the starting offset addresses, the data contents of that location and the contents of the consecutive memory locations upto the end address, if an end address is specified.

   The line format is arranged such that any subsequent lines, (if present) will begin with an offset address whose last nibble is zero. A line consists of a maximum nunber of 16 bytes or 8 words. In case of stand-alone operation the same amount of data as contained in one-line is displayed on the LCD making use of all the 4 lines available.

3. The display memory commands can be cancelled, or the output can be stopped and resumed at any time by entering appropriate control parameters

During Serial operation, entering **Ctrl+C** at any time immediately terminates the command.Monitor returns to the command entry mode. **Ctrl+S** stops the output but does not terminate the command. Entering **Ctrl+Q** now resumes the output from the point where the output is stopped while **Ctrl+C** will terminate the command.
Note that **Ctrl+S** should be followed only by **Ctrl+Q** or **Ctrl+C**. no other console input is allowed following **Ctrl+S.**

During stand-alone operation, the monitor waits for a user strobe viz. <CR> to refresh and display the next set of data. Pressing <Esc> instead will terminate the command and the monitor returns to command entry mode.

ERROR CONDITONS:

1. Specifying an end address that is less than the offset value of the start address.

EXAMPLES

1. Display contents of location 140H relative to DS register. Assume DS is set to 0000.

```
.D DS: 140 <CR>
0000:0140  A0
```

2. Display contents of location 100:10CH through 100:125H.

```
.D100:10C, 125 <CR>
                00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F

0100:010C                                                       F0   00   00  9C
0100:0110       EA  BD  04  BA  00  F2  10  21  A4  35  14 17 FA  F0  F1  F5
0100:0120       30   31   32  34   35
```

In addition to the above. the ASCII characters equivalent to the data bytes are also displayed on the right hand side of each line of output only in serial mode of operation. The non-existent ASCII equivalent data is represented by a '.'.

In the stand-alone mode the output for the above command will be in the format described below

```
.D100:10C, 125  <CR>
0100:010C
F0  00  00  9C
```

The monitor now waits for a user stobe viz. <CR> to refresh the display and output subsequent data

```
<CR>
0100:0110
  EA  BA  04  BA  00  F2
  10   21   A4  35   14  17
   FA  F0  F1   F5
<CR>
0100:0120
   30  31  32  32   34   35
<CR>
   .
```

3. Dispay word at location 10C H relative to DS register. Assume DS is initialized to 200.

```
.DW DS : 10C<CR>
                  0000    0002    0004    0006    0008    000A   000C   000E

0200:0100                                                        00F0
```

In case of stand-alone operaaton, the LCD will output

```
.DW DS : 10C<CR>
0200:010C
00F0
<CR>
.
```

For displaying a range of data words the same procedure as with display byte is followed, with the exeption that **DW** replaces **D.** Note that ASCII equivalent characters are not diaplayed with word data.

4.5.3 EXAMINE/MODIFY REGISTER COMMAD

FUNCTION

The Examine/Modify register (x) commad is used to examine and optionally modify the contents of any of the 808/8088's registers.

SYNTAX

X[<reg><CR>[<new data>/,]] <CR>

OPERATION

1. To use the Examine/Modify register command, enter X when prompted for command entery.
2. If the current contents of all the registers are to be examined, enter a carriage retun. Now the monitor will output the contents of all 14 registers.
3. If the conents of a particular register are to be examined or modified, enter the abbreviated register name after entering X and press <CR>. (The abbreviations for the register names are shown in the table 4.2)

Table 4.2 register abbrevations

Register Name	Abbreviation
Accumulator	AX
Base	BX
Count	CX
Data	DX
Stack pointer	SP
Base pointer	BP
Source index	SI
Code segment	CS
Data segment	DS
Stack segment	SS
Extra segment	ES
Instruction pointer	IP
Flag Register	FR
Program counter*	PC
Destination Index	DI

Now the monitor will output an equals sign '=', the current contents of the specified register, the data prompt character '-' and a space. If you wish to change the contents of this register, , enter the new value following by a comma, or carriage return.

Enter a carriage return terminates the command. Entering a comma displays the contents of the next "sequential"register and opens it for optional modification. the register sequence is in the order as shown in the above table. This sequence is not circular i.e. if a comma is entered after the contents of the "last modifiable" register (viz. flagesregister FL) is examined/modified, the command is terminated and the monitor returns to the command entry mode.

**Program counter cannot be modified using this command.*

EXAMPLE

1. Examine the contents of all the registers

.X <CR>
AX=1182 BX=A113 CX=00F DX=1242 SP=0100 BP=4020 SI=6020 DI=6F20
DS=0000 SS=0000 ES=0000 SC=0000 IP=0200 FL=F046 PC=0000

In case of stand-alone mode operation, the display format will be

.X <CR>

AX=1182 BX=A113
CX=000F DX=1242
SP=0100 BP=4020
SI=6020 DI=6F20

The monitor now waits for a user strobe viz. <CR> to refresh the display and output subsequent data

DS=0000 SS=0000
ES=0000 CS=0000
IP=0200 FL=F046
PC=0000
<CR>
.

2. Examine and modify the SP register and examine the next register i.e., BP

.XSP <CR>
.XSP=0100 - 0110,
BP=4020-<CR>
.

4.5.4 MOVE MEMORY COMMAD

FUNCTION

Move memory command is used to move a block of data from one area of the memory to another area.

SYNTAX

M<start address>, <end address>, <destination address><CR>

OPERATION

1. To use the move command, enter M when prompt for command entry , then enter the three required paramenters separated by commas.
   The three paramenters are the "start" and "end" address of the memory block to be moved and the start address of the destination block . In this command, the end address is relative to the segment value specified with 'start address' or implied with the 'start address' (i.e. the current contents of CS register if no segment value is specified). Consequently, no segment value is permitted with the 'end address' and with each command execution, a maximum of 64K bytes can be moved. The segment for the destination address is relative to the current CS value unless otherwise specified.

   After entering the three paramenters enter carriage return.

2. Now the monitor moves the contents of the memory block from "start address" to "end address", of consecutive memory location beginning at 'destination address'.

   After moving the block of memory, monitor returns to the command entry mode.

Note : SInce move operation is performed one byte at a time, M command can be used to fill a block of memory witha predefined constant. To do this move the constant into the "start address" location using S command, then use M command with a "destination address" which is one greater than the "start address'. Now the memory block from start address to "end address" +1 is filled with the value contained in "start address".

ERROR CONDITIONS:

1 Trying to move data into a non- existent or read-only (e.g. ROM or PROM) memory location or outside the user RAM.

2. Specifiying an 'end address' value that is less than the offset value of the "start addresss"

EXAMPLES :

1. Move the contents of the locations 2000H through 2020H relative to CS register, to the memory block starting at 3000H relative to the DS register.

   .M 2000,2020:2000, DS:3000 <CR>

2. Fill memory location 1000:2000H through 1000:2100H with constant AAH.

   .S 1000:2000, <CR>
   1000:2000 BA-AA <CR>

   .M 1000:2000,20FF,1000:2001<CR>

The validity of the Move Memory commands can be verified anytine usiing the Display Memory or compare memory commands.

4.5.5 FILL BYTE AND FILL WORD COMMADS

FUNCTION

This command is used to fill a block of memory with constant byte or word data.

SYNTAX

F<start address>,<end address>, <byte value><CR>
FW<start address>,<end address>,<word value><CR>

OPERATION

1 Both commads operate in a similar way, Enter F (to fill byte) or FW(to fill word) when prompted for command entry then the starting address of the block of memory to be filled followed by comma and ending address of the block.

2 After entering the address parameters, a comma followed by the byte or word data should be entered. Pressing <CR> now executes the command and the monitor returns to the command entry mode.

3. The user may check the validity of this command by using Display MEmory command as described above.

Note: The command will accept the last two Hex characters, as vaild data in case of byte operation and the last four valid Hex characters as word data during the respective operation. If less than four characters are entered in case of word operation the commad assumes the upper nibbles not entered to be 0.

EXAMPLES

1. Filling the memory locations from 0:2000 to 0:2050 with constant data byte 55
F0:2000,2050,55<CR>

2. Filling the memory locations from 0:5000 to 0:6000 with constant word data 1234
FW0:5000,6000,1234<CR>

3. Filling the memory location from 0:5000 to 0:6000 with constant word data 00AA
FW0:5000,6000, AA<CR>

ERROR CONDITIONS

1. Entering the ending address offset lesser than the starting address offset

2. Trying to enter a non-Hex value for the address or data.

3. Trying to fill non-existent memory or ROM locations.

4.5.6 INPUT BYTE AND INPUT WORD COMMADS

FUNCTION

The Input byte (I) and input word (IW) commands are used to input (accept) a byte or word from an input port, and to display the accepted byte or word.

SYNTAX

I<Port Address><CR>[,]*<CR>

IW<Port Address><CR> [,] *<CR>

OPERATION

1. Both the commands operate in a similar way. Enter I (to input byte) or IW (to input word) when prompted for command entry and then enter the address of the port to be read. Since the I/O space is only 64K bytes, no segmet value is permitted with 'port address'

2 After entering port address and <CR>, enter another <CR> or a comma. The monitor reads the bytes or word at the specified port and displays it on the console.

3. Each subsequent comma incerements the port address and displays the current data at the addressed port on a new line. A carriage return terminates the command and the monitor returns to the command entry mode.

EXAMPLES:

1 Read a byte I/O port at address FFE1H

.IFFE1 <CR>
FFE1 - FA <CR>
.

2 Read a series of word from I/O ports located at addresses FFE0H and FFE6H.
(Note that when using word input i.e. IW command, lower order address is entered as port address)

.IWFFE0 <CR>
FFE0 - A2A2,
FFE2 - B2B2,
FFE4 - C2C2,
FFE6 - D2D2 <CR>

.

4.5.7 OUTPUT BYTE AND OUTPUT WORD COMMANDS

FUNCTION

The output byte (O) and output word (OW) commands are used to write a byte or word to an output port.

SYNTAX

O<port address><CR>[<data>/,]*<CR>

OW<port address><CR>[<data>/,]*<CR>

OPERATION

1. Both the output commands (O and OW) operate similarly. Enter O to write a byte, or OW to write a word when prompted for a command entry and then enter the address of the port to be written followed by a <CR>. No segment value is permitted with the port address. (I/O address spece is limited to 64K bytes).

2. After this, enter the data to be written in Byte or Word format as applicable or a comma if you wish to skip writing data at this port. Entering a comman will increment the port address after which the monitor displays the port address on a new line and the process continues.

3. If the user enters a data byte or word as applicable, a comma or <CR> should follow it.

Entering a carriage return sends the data to the specified port, terminates the command and returns monitor to the command entry mode. Entering a comma sends the data to the specified port but does not terminate the command, and it permits subsequent data output to I/O ports in succession. The user will always have and option to either output data at a port or skip the port and proceed with the next subsequent I/O port. At any stage a carriage return terminates the command.

EXAMPLES

1 Write to the I/O port FFE6H for configuring it as output port by sending appropiate command byte to command register FFE6H.

.OFFE6 <CR>
FFE6 -80<CR>

2. Output a series of words to ports ranging from FFE0 to FFE6

.OFFE0<CR>
FFE0 - 1234,
FFE2 -ABCD,
FFE4 - 5555,
FFE6 - 3676 <CR>
.

4.5.8 COMPARE MEMORY COMMAND

FUNCTION

This command compares a block of memory with another black and displays differences in the location when found.

SYNTAX

C<start address 1 >,<end address 1>, <start address 2> <CR>

OPERATION

1 To use this command enter C when prompt for command entry and then enter the starting and ending addresses of the first memory block separated by a comma. Followed by a comma and the starting address of the second memory block should b entered.

NOTE : In this command, **end address 1** is always relative to the segment value specified with **start address 1** or implied with **start address 1** (i.e. the contents of CS register if no Segment value is specified). Consequently, no segment value is permitted with the **end address1** and with each command

execution; a maximum of 64K bytes can be compared. The segment for **start address2** is relative to the current CS value unless otherwise specified.

2. Pressing <CR> starts exeution of the command. The monitor will compare subsequent data at each corresponding location of the blocks and display differences when found in the following pattern. The display format for stand-alone mode of operation is described below

   Segment 1:offset1 - data segment 2:offset2 -data

3. The monitor returns to command entry mode at the end of command execution

EXEAMPLES

1. Compare a block of memory beginning at 0:2500 to 0:2050 with a block beginning at 0:3000.

   C0:2000, 2050, 0:3000<CR>

   .

   The result shows that there are no mismatches.

2. Compare a block of memory beginning at 0:2500 to 0:2550 with a block beginning at 0:3000

   . C0:2500, 2550,0:3000<CR>

0000:2505 - 45	0000:3005 - 76
0000:2515 - 21	0000:3015 - 16
0000:2532 - 45	0000:3032 - A3
0000:2544 - 1A	0000:3044 - 22

   .

The result shows the memory mismatch at 4 locations.

In case of stand-alone operation, the command entry format is similar to that in the serial mode. The output in case of example 2 will be as follows.

```
0000:2505 - 45
0000:3005 - 76 <CR>
0000:2515 - 21
0000:3015 - 16 <CR>
```

The display will now be refreshed and will output the following result.

```
0000:2532 - 45
0000:3032 - A3 <CR>
0000:2544 - 1A
0000:3044 - 22 <CR>
.
```

ERROR CONDITIONS

1 Entering the ending address offset lesser than the starting address offset

2 Trying to enter a non-Hex value for the address or data.

3. Entering a value for the offset of the second address such that the range of locations left in the segment from this offset does not match the range to be compared.

4.5.9 GO (Execution) COMMAND

FUNCTION

Go (G) Command is used to transfer CPU Exeution from the Monitor t user program.

SYNTAX

G<CR>[<start address>[, <breakpoint address>]] <CR>

OPERATION

1. To use this command enter G when prompted for command entry followed by a <CR>. The monitor now outputs the current CS and IP register contents in the **Segment:Offset** format.
2. If the user wishes to start program execution from an alternate starting address, the user has to enter the same. Segment specification is not necessary if the segment value of the program start address is the current code segment.
3. The user may use the breakpoint facility by entering a comma followed by a breakpoint address at which program execution needs to be stopped. no segment value is permitted with breakpoint address. Thus the default segment value for the breakpoint address. Thus the default segment for the breakpoint address is either the segment specified with the "start address" or the current CS register content if a segment value is not specified.
4. To transfer the cpu Exceution from monitor program to user program, enter the user program starting address as mentioned in the Step1. User program termination is entirely dependent on the instructions entered by the user.

5 To exit from the user program execution and to return to monitor program the user has the following options.

- At the logical termination of the user program the user can enter the instrcution INT3 (Hex code: CC). This instruction will execution the Type 3 interrupt service routine embedded in the ESA 86/88E. Monitor, which will break the program at that address and displays all the current register contents and returns to command entry prompt.
- If the user program ends with a HLT instruction (Hex Code =F4) or continues to execute in an endless loop, the user may have to RESET the trainer. However upon RESET,any previous information of the program contained in segment and status registers will be lost and they are reinitialized to their default values.
- The user can use the breakpoint facility provided with the GO command and can break the program at any desired address. With this option, the user program breaks at the specified address and control is transferred back to the monitor. The details about breakpoint facility are discussed in following section.

4.5.10 BREAKPOINT FACILTY

GO command permits the optional specification of "breakpoint address" When program reaches the breakpoint address, control is transferred to the monitor and contents of all registers are saved. Before returning to command entry prompt, the monitor displays the message + BR@ following by a display of all processor registers, the breakpoint address in **segment:Offset** format, the machine code and disassembled line of the instruction at the break point address.

If a subsequent GO command is entered, the current address displayed will be the breakpoint address. The user can resume execution of the program from this address by just entering<CR>

Upon receiving GO command with a breakpoint address, the Monitor program saves the instruction at the breakpoint address, and replaces it with an interrupt instruction before transferring control to the user program . When the program reaches the breakpoint address, control is returned to the monitor. The monitor saves all CPU registers, restores the break-point instruction and output the previously described pattern. It then issues the command entry prompt allowing the user to modify any of the registers, memory locations before resuming the execution of the program.

From the above discussion, it is clear that

- The user can specify a breakpoint address only when a comma preccedes it.
- The segment value for the breakpoint address is always relative to the "start address" segment value.
- An instruction in the read-only memory cannot be break-pointed.
- Breakpoint address must be specified each time when a program is to be broken at a desired location.

+BR@ is displayed on the console i.e. it appears only in the serial mode. In stand-alone mode the monitor procceds with the register display directly.

EXAMPLES

1. Transfer control to the program at 2000H relative to CS register

   .G<CR>
   .G 0000:0000 2000<CR>
   .

2. Transfer CPU control to a program starting at location 20:300H and break the instruction at lication
   20:3F2H

   .G<CR>
   .G 0000:0000 20:300, 3F2<CR>

```
        BR@
AX=1182   BX=A113   CX=000F   DX=1242   SP=0100   BP=4020   SI=6020
DI=6F20   DS=0000   SS=0000   ES=0000   CS=0020   IP=03F2   FL=F002
PC=005F2
        0020:03F2    55                          PUSH  BP
        .
```

In Stand-alone mode of operation, the output format on the LCD with respect to the above command will be as described below.

```
.G<CR>
.G  0000:0000
20:300 , 3F2<CR>
```

The monitor refreshes the LCD and displays all the CPU registers in the following format. In this mode the monitor waits for user strobes wherever necessary.

```
AX=1182       BX=A113
CX=000F       DX=1242
SP=0100       BP=4020
SI=6020       DI=6F20
<CR>

DS=0000       SS=0000
ES=0000       CS=0020
IP=03F2       FL=F002
PC=005F2<CR>
0020 : 3F2
PUSH  BP
55
<CR>
```

3. Attempting to insert a breakpoint in the EPROM area. As shown below, this will result in an error.

```
.G<CR>
.G  0000:0000  F000:100, 122<CR>
?G 0000:0000  F000:100,  122
.
```

NOTE: All underlined lines in these example are either overwritten by the statements following or are inserted in the text merely for illustration.

4.5.11 SINGLE STEP COMMAND

FUNCTION

Single step (N) command is used to excute single instruction of a program. After the excution of single instruction, CPU control is returned to the monitor from the program being executed.

SYNTAX

```
N<CR>[<start address>],[,/<new address>]* <CR>
```

OPERATION

1. To use the single step command, enter N and with <CR> when prompted for command entry.
2. The monitor will display the current contents of CS & IP registers in **Segment : Offset** format.
3. If the execution of an instruction at an address other than the displayed address is required, the user has to enter a "start address". The segment value of the start address is implied to be the same as the current value of the CS register unless it is explicity entered.
4. This has to be followed by a comma. the monitor now outputs the current content of the CPU registers following by line of disassembled code at the start address. To procced with execution of this instruction,another comma has to be entered.
5. After executing the addressed instruction, the monitor saves and displays all the register contents. The monitor also displays a line of disassembled code of the instruction at the next subsequent address. Now the user can enter a comma and proceed with executing the instruction at the displayed address or enter a new address at which an instruction is to be executed. Note that the monitor does not output a command entry prompt, even though the control rests with the monitor. Hence entry of any other command at this stage will result in an error.
6. A carriage return, after executing an instruction terminate the command and returns the monitor to command entry mode. Modification of any memory or register contents can be done whike single stepping through a program only by using a <CR>. However since the current command is terminated, the user will have to re-enter the single-step command and proceed.

RESTRICTIONS

1. An insturction that is part of a sequence of instructions that switches between stack sigments (i.e.changes the SS and SP register contents) cannot be excuted using single step cmmand.
2. If an interrupt occurs, while executing an instruction using single step command or if the execution of that instruction generates and interrupt, then CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (Breakpoint) interrupt insturction (CCH) should not be executed using single-step commad (since its execution will cause the program execution sequence to enter into the Monitor Program).

EXAMPLES

1. Single stepping a program from location 0000:2000H onwards,The program considered here adds a value of 15h to the accummulator contents and adjusts, the result to obtain the BCD value.

N<CR>
N 0000:0000 2000,

```
AX=0015   BX=0000   CX=0000   DX=0000   SP=0100    BP=0000   SI=0000
DI= 0000   DS=0000   SS=0000   ES=0000   CS=0000    IP=2000    FL=0000
PC=02000  0000:2000   05 15 00    ADDW     AX,0015
/
BR@

AX=002A  BX=0000  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000
DI=0000
DS=0000  SS=0000  ES=0000  CS=0000  IP=2003  FL=FOO2  PC=02003
0000:2003  27                  DAA
/


BR@
AX=0030  BX=0000  CX=0000  DX=0000  SP=0100  BP=0000  SI=0000
DI=0000
DS=0000  SS=0000  ES=0000 CS=0000 IP=2003  FL=F002  PC=02003
0000:2004  CC                  INT 03
<CR>
.
```

For further illustration of this command, please execute the example program given in Chapter 10 using step command and observe the results.

4.5. 12 HELP COMMAND(H)

FUNCTION

This command lists of all ESA 86/88E Monitor commands with their valid syntax in stand-alone mode of trainer operation.

H[<command mnemonic>] <CR>

1. When prompt for a command entry, entering H with <CR>list ESA 86/88E monitor command with a description of their valid syntax on the LCD. Thus the user can use this command as ready reference source on the monitor commands.

2. If the command syntax of a particular command is required then the user should enter H followed by the command mnemionic. For e.g. entering HS with <CR> will output the syntax of the Substitute memory command.

After displaying the command, the monitor waits for a user strobe viz. <CR> to display the next command and its syntax. Pressing <Esc> will terminate the command and the monitor returns to command entery mode.

The command syntax appearing with the description of this command is the same as that of the command Help appearing in serial Mode of Operation.

4.6 MORE MONITOR COMMANDS

1. Refer chapte 7 of this manual for a detailed description of ESA 86/88E EPROM programmer system invoked by the following command.

P (invoke Programmer) **P <CR>**

The EPROM Programmer software supports commands to Read,Blank check, Verify and/or Program the popular EPROM 2716 through 27512 using the ESA EPROM Programmer interface, which may be optionally obtained from the manufacturer.

2. Refer chapter 8 of this manual for a detailed description of ESA 86/88E Symbolic one-line Assembler and a discussion of the following monitor commands in support of this assembler.

A (invoke Assembler) **A[address>] <CR>**

LL (list lables) **LL<CR>**

LC (Clear labels) **LC<CR>**

Z(Disassembly) **Z[<start address>[<end address>]]<CR>**

This chapter described the use of most of the commands of ESA 86/88E monitor. It is urged that the user go through this chapter carefull to become familiar with the ESA 86/88E system. Commands related to the EPROM PROGRAMMER are discussed in chapter 7 while those related to the assembler and disaddembly are duscussed in chapter 8.

CHAPTER 5

SYSTEM DISIGN DETAILS AND MEMORY ADDRESSING

This chapter gives an insight into the hardware design details of ESA 86/88E Microprocessor trainer. The onboard interfacing options and external interfaces supported by the trainer are discussed in subsequent chapters.

Appendix A gives the complete schematics, Appendix B Gives the connector details and Appendix C has the component layout diagram.

The design details discussed in this chapter include.

a) CPU
b) Memory addressing
c) System timing
d) Interrupt system
e) 8087 Co-processor(NDP)
f) Bus expansion

5.1 CPU

ESA 86/88E can be operated with either the 8086 CPU or 8088. A single DIP switch setting as explained earlier in the manual can make the appropriate processor selection. In either case, the CPU is configured for maximum mode of operatoon.

5.1.1 CPU RESET

An on-board RESET key provides the reset signal to the CPU through the 8284 A -clock generator. The RESET Key gives an RSTIN* signal to the 8284 and its RESET output is used to reset the CPU and the rest of the system synchronously. this signal is available on the bus connector also and can be used to reset off-board peripherals.

5.1.2 CPU CLOCK

A 15MHz crystal is the clock source for 8284A-clock generator. The 8284A divides this frequency by three and produces a 5MHz clock with 33% duty cycle as required by8086/8088. Further,8284A Provides a 2.5MHz PCLK with 50% duty cycle, which can be used an a clock input for onboard peripherals. the PCLK is also available on the bus connector and may be used as clock source is external peripkerals.

5.1.3 CPU ADDRESS BUS

Latches (74LS373s) at U 16, U23 and U24 are used to latch the addresses with the help of ALE signal. These latches are always enabled since there is no provision for external bus master. In addition to these BHE* , status signals S0*, S1*, S2* are also latched.

As the CPU operates in maximum mode, the 8288 Bus controller is used to decode the status signals and to provide all the control signals.

5.1.4 CORRECTED BHE*

8086 processor generates a BHE * signal on pin 34 and this signal is used to access the devices residing on the upper data bus. However, 8088 Processor always drives a high signal on pin 34 as its access is always over the lower data bus. To accommodate this difference, a corrected BHE* (CBHE*) signal is used in rest of the system. This corrected BHE* signal is either the latched BHE* signal (for 8086) or the inverted latched A0 signal (for 8088).

The setting of SW7 of the DIP Switch indicates the selection of 8086/8088 CPU and the enable signals for the corresponding CPU are derived from the decoding IC at U5. if 8086 is selected the latched , the IC will pass the inverted A0 signal as corrected BHE * Signal.

5.1.5 CPU DATA BUS

BI-directional buffers (74lS245 s) at U 13,U14 and U15 are used to buffer the CPU data bus. One of these buffer (741S245 at U15) is used as swap buffer to route the upper data bus to lower data bus when 8088 is selected. The enable signal for this bus are derived from CBHE*, AO, DEN * (from 8288 bus controller) and 86SEL* signals. the Logic implemented by U5 as follows.

CPU	DEN*	A0	CORRECTED BHE*	ENABLE FOR		
				Upper Data Bus Buffer	Lower data bus buffer	Swap Buffer
8086	HIGH	X	X	NO	NO	NO
	LOW	LOW	LOW	YES	YES	NO
	LOW	LOW	HIGH	NO	YES	NO
	LOW	LOW	HIGH	YES	NO	NO
8088	HIGH	X	X	NO	NO	NO
	LOW	LOW	HIGH	NO	YES	NO
	LOW	HIGH	LOW	NO	NO	YES

(X:Don't care : the combinations not shown above never occur)

5.2 SYSTEM TIMING

The time-base for CPU operation is derived from an 8284 A Clock Generator. The CPU may be operated at a frequency of 5MHz, which is the clock output from the clock generator with 33% duty cycle. the clock generator also generates clock output PCLK with 50% duty cycle. This clock frequency is 2.5 Mhz when the 8284 input is from the 15MHz crystal. The 8284 clock generator also generates synchronous Ready and Reset signals to the CPU.

The serial interface controller 8251A USART user a separate crystal of 6.144 MHz as its clock frequency source. the clock dividing circuit implemented using 7474 IC at U19 gives a clock frequency of 3.072 MHz to the USART and 1.536 Mhz to timer 0 of 8253, which is used to generate the Baud Clock.

5.3 MEMORY ADDRESSING

ESA 86/88E has four 28-pin JEDEC compatible slots labeled as U 17 -U18 & U25-U26 for memory ICs Sockets U17 & U25 are populated with system firmware using EPROMs 27256 (32Kx2=64K) or 27512 (64K x2 =128K). Sockets U18 & U26 are populated with static RAMs 62256 (32kx2=64K).

EPROM configuration using jumper setting is described in chapter 2. However the RAM area starts from 0:0000H, the memory from 0:000H to 0:1 fffh is used by the system for interrupt Vectors, Stack and Assembler data tables Thus user RAM starts from 0:2000H

5.3.1 MEMORTY MAP

Memory Type	Sockets Used	Device	Address range (in Hex)
EPROM	U17 & U25	27256	F0000 - FFFFF
EPROM	U17 & U25	27512	E0000 - FFFFF
RAM	U18 & U26	62256	00000 - 0FFFF

Optional battery backup provision is available for RAM using an onboard 3.6V Ni-Cd. cell. Refer Appendix C for Component layout diagram.

5.4 INTERRUPT SYSTEM

Hardware Interrupts: the 8086/8088 CPU has two hardware interrupts, called NMI and INTR.

- The NMI or type 2 interrput is connected to the KBINT key on the trainer. Providing the vectoring information for this interrupt is entirely left to the user. To make use of this interrupt, the user should enter the instruction pointer word of the Interrupt Service Routine at address 0:0008H and the code segment word at address 0:000AH. Then, pressing the NMI service routine address specified by the user. An example using the KBINT key is given in chapter 10.

- The INTR (maskable interrupt) line is available to user on system expansion bus connector J6 for external inter face.

Internal Interrupts : INT 3 can be used by user programs to return the control to ESA 86/88E monitor. INT 1 is reserved by the Monitor and is used for Single stepping. Other internal interrupts are available to user.

5.5 8087 NUMERIC DATA PROCESSOR

On-board provision is made for 8087 co-processor. The CPU can be either 8086 or 8088 and to install 8087 no hardware change are necessary. Switch off power to trainer and populate socket U21 with 8087 IC.
Refer the component layout diagram in Appendix C and 8087 Instruction set in Appendix F.

5.6 BUS EXPANSION

The buffered address, data and control signals are provided on two 26-pin bus connectors (i.e J6 & J7) as given below. The user may use this information and implememt bus interfacing accordin to specific needs. The signal definitions and details of all interfacing connectors and headers, including bus expansion are described in Appendix B.

J6 ADDRESS & INTERRUPT LINES

PIN NO.	SIGNAL	PIN NO	SIGNAL
1	MRCC#	2	MWTC#
3	-	4	BA 19
5	INTR	6	BA 18
7	BD15	8	BA 17
9	BA15	10	BA 16
11	BA13	12	BA 14
13	BA11	14	BA 12
15	BA9	16	BA 10
17	VCC	18	VCC
19	BA7	20	BA5
21	BA4	22	BA3
23	BA0	24	BA8
25	BA2	26	BA1

J7 DATA & CONTROL LINES

PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	ALE	2	BD 9
3	BD8	4	BD 10
5	CBHE#	6	BD 11
7	-	8	BD 12
9	BA 6	10	PCLK
11	BD 13	12	RESET
13	AIOWC	14	BD 14
15	IORC#	16	INTA#
17	GND	18	GND
19	BD1	20	BD 0
21	BD3	22	BD 2
23	BD5	24	BD 4
25	BD7	26	BD 6

Note : '-' indicates Noz Connection

CHAPTER 6

ONBOARD PERIPHERALS

ESA 86/88E Microprocessor Trainer features several onboard peripherals for a wide range of user applications, some of which are standard features of all ESA Trainers. This chapter introduces the user to the following onboard peripheral interfaces and their applications along with the I/O address map. This chapter also includes peripheral details and summary of ESA 86/88E Trainer connectors.

- 8255A Programmable Peripheral Interface
- 8253 Programmable Interval Timer
- 8251 A USART
- 8042 Universal Peripheral Interface
- LCD Interface
- Parallel Printer Interface.

6.1 I/O ADDRESSING AND I/O MAP

I/O decoding is implemented using the decoding IC at U9. The following table summarizes the I/O mapped I/O address assignments and usage of all ESA 86/88E peripherals.
(Refer Appendix C for Component Layout).

PERIPHERAL	PORT ADDR	DEVICE RESGISTER	USAGE
8251A USART	FFF2	Command/status port	Serial via
at U10	FFF0	Data port	RS-232-C interface.
8255-1(PPI)	FFE7	Control Port	
High, at U20	FFE1	High port A	24 I/O lines available to
	FFE3	High port B	user for interfacing.
	FFE5	High port C	
8255-2(PPI) to	FFE6	Control Port	24 I/O lines available
Low, at U1	FFE0	Low port A	user for interfacing
	FFE2	Low port B	and parallel
	FFE4	Low port C	Printer Interfacing
8253 (PIT)	FFFF	Command Port	Timer 0 used for Baud clock
at U8	FFF9	TIMER 0	generation.
	FFFB	TIMER 21	Timer 1 and 2 are
	FFFD	TIMER 2	available to user.
8255-3 at U2	FFDE	Control Port	
	FFD8	Port A	I/O lines used for interfacing
	FFDA	Port B	LCD and to read DIP switch.

	FFDC	Port C	
8042 UPI at U3	FFEB	Command Port	Interfacing with PC keyboard
	FFE9	Data Port	in stand-alone mode operation

6.2 PROGRAMMABLE PERIPHERAL INTERFACES

ESA 86/88E Comprises of three onboard 8255A Programmable Peripheral Interface (PPI) devices.Of these. two 8255s placed at U1 and U20 provide 48 I/O line that are entirely available to the user The peripheral at position U20, resides on the upper data bus, while the peripheral at U1 reside on the lower data bus. Each 8255A consists of three 8-bit input/output data ports(designated as port A,B, and C) and one write-only control port. All the port can be address individually and configured independently input or output mode.

The I/O address assignment can be found in the I/O map given in the earlier section.

The signals of the port A , port B and Port C of the 8255A at UI are brought out to the connector J4 and the port lines of other 8255A at U20 are brought out to the connector J5. The pin assignment for the individual port lines are given in Appendix B.

6.3 PROGRAMMABLE INTERVAL TIMER

ESA 86/88E has an onboard Programmable Interval Timer 8253 positioned at U8. 8253 has one command port and three data ports viz . Timer 0,1 and 2, which provided three independent timer/ counter channels. Of these, timer 0 is initialized by trainer firmware for gernerating appropriate baud clock depending upon DIPswitch setting for serial mode of operation. This output is fed to the transmit and receive clock pin of the 8251A USART.

The other two timers are made available to the user. The signals of timer 1,and 2 are available on the header J9. Refer to the connector details in Appendix B. The I/O address assignments can be found in the earlier section.

6.4 SERIAL INTERFACE CONTROLLER

Serial communication in ESA 86/88E microprocessor trainer is implemented using 8251A USART. The trainer firmware initializes the 8251A for full duplex asynchronous operation, 2 stop bits, No parity check and baud rate factor of 16x clock. The clock for 8251 A is generated using a 6.144 MHz crystal. The Baud clock or Transmit/Receive clock is derived from Timer 0 of 8253, as described earlier.

ESA 86/88E Trainer supports RS-232-C serial communication standard. A +5V powered RS-232 driver/receiver generates the standard voltage levels. I/O address assignments of the8251A registers are given earlier in this chapter.

6.5 8042 UNIVERSAL PERIPHERAL INTERFACE

ESA 86/88E Trainer support can be interfaced with a PC ASCII keyboard for stand-alone mode of operation. The keyboard interface is controlled by an 8042 Universal Peripheral Interface microcomputer. The UPI is a general-purpose device that allows the user to develop customized solutions for peripheral device control.

The I/O addresses for 8042 UPI are given earlier in this chapter. The UPI uses clock

inputs from the PCLK output of the clock generator 8284. The command/status port of 8042 gives the status of keyboard input. The keyboard input can be read by polling the status of output buffer register. The keyboard sends scan codes for the respective keys pressed. The scan codes for the keys can be referred in the PC AT reference manual. The UPI 8042 firmware supports either 71 or 84 keys PC keyboards.

6.6 LCD INTERFACE

A 20x4 LCD is provided on the trainer as display terminal. The LCD is used as console display by the trainer firmware for stand-alone mode of the trainer.

The LCD module has
Two registers

- Instruction Register
- Data Register

Three control signals RS, R/W and E which selects the operation of LCD.

E	=	1	:	For any operation with the LCD
RS	=	1	:	Operation with Data Register
RS	=	0	:	Operation with InstructionRegister
RW	=	0	:	Write to LCD
RW	=	1	:	Read from LCD

The LCD is interfaced to the port lines of 8255 at U2. The trainer firmware uses thesse port line for initialization and display. These line are terminated on a flow-strip connector J3, to which the LCD is connected. An example program, which initializes and displays some characters on LCD, is given in chapter 10.

6.7 PARALLEL PRINTER INTERFACE

ESA 86/88E trainer is facilitated with Centronics compatible parallel printer interface for serial mode of operation. The printer interface can be enabled by the DIP switch (SW5) on the trainer to obtain hard copy of the operations performed in serial mode of the trainer.
80/132 column printers are recommended for this interface to get the proper formatted printouts.

6.7.1 CONFIGURATION AND OPERATION

- Switch OFF the power to trainer
- Configure the trainer to serial mode of operation
- Enable printer driver using onboard DIP switch SW5 by switching it ON.
- Connect the 26-pin end of the printer cable to J4 of ESA 86/88E and the other end to the printer.
- Switch ON the power to trainer as well as the printer. Check for proper sign-on message as described in chapter 2.

- Now if the command prompt does not appear for serial mode of trainer then check the configuration settings, the connector cable and Online/ready indicator on the printer.
- When the command prompt appears, the printer is assumed to be attached to ESA 86/88E
- To detach or disable the printer, switch off SW5 of on-board DIP switch and press RESET.

NOTE : The printer cable may be obtained from ESA as an optional item. The user may build a suitable connector using the details given in sections 6.7.6 and 6.7.7. However note that the cable must be short enough to be driven by 8255 port line. A maximum length of 3ft is recommended for reliable operation.

When the printer is attached and enabled, any character output to the console display will also be written to the printer. For example to obtain a hard copy of the contents of memory location in Hex, (byte format), use the Display command **D<addr1>,<addr2> <CR>.** The entire contents of memory locations between the specified range will be printed exactly as it appears on the console display, including the command line.

NOTE : All control and invalid ASCII characters will be printed as "."

Similarly user can obtain a disassembled listing of any program by using the Z command.

6.7.2 THEORY OF OPERATION

The printer driver make use of 8255 (at U1) port lines which are terminated on the connector J4. The printer driver in the trainer initializes the 8255 ports as given below, if the driver is enabled using SW5 of DIP switch.
Port A as Input port and Port B, Port C as output ports. The ***STROBE**** and ***BUSY*** signals which are
connected to the port lines are used as handshaking signals.

NOTE: The user can use this 8255 as general purpose I/O, When the printer driver is not enabled i.e.when SW5 of DIP switch is OFF.

6.7.3 ERROR MESSAGES

- If the printer dose not assert the ***Busy*** line then the trainer will display the message ***"Printer Busy....Waiting"*** . The status of busy line can be kept under polling, until it is asserted. This process can be aborted by pressing **CTRL+C** the trainer will then display the message ***"Print aborted".***
- If no acknowledgement for data transfer is received from printer then a message ***"No ACK from Printer"*** will be displayed on the console.

6.7.4 PORT SPECIFICATIONS

The signals used for Parallel Printer interface conform to the following spec.

- Centronics Compatible
- Synchronization via STROBLE* pulses
- Handshaking via BUSY signal
- logic levels TTL compatible

6.7.5 CONNECTOR DETAILS

J4 Pin Details (8255 Port s)	Signal Name from 86/88E	Input(or) Output	Description	Centronics Connector Pin Details
5 (PCO)	***STROBE****	O/P	STROBE* pulse to the printer	1
16(PA7)	***BUSY***	I/P	A high on this indicates that printer is busy.	11
13(PBO)	Data 0	O/P	These signals	2
14(PB1)	Data 1	O/P	represent 8bits	3
11(PB2)	Data 2	O/P	of parallel data	4
12(PB3)	Data 3	O/P	High =1	5
9(PB4)	Data 4	O/P	Low =0	6
10(PB5)	Data 5	O/P		7
7(PB6)	Data 6	O/P		8
8(PB7)	Data 7	O/P		9
26	GND		Signal ground	19

[1]The signal become high in one of the following cases

a) During data entry
b) During Printing Operation
c) In OFF-LINE states
d) During printer error status

6.7.6 DIRECT OUTPUT TO PRINTER

As described above, when the printer interface is enabled, any character sent to the console will also be sent to the printer. This facility is available in the serial mode of opreation only. However, the trainer firmware provides a ***"Print String"*** routine to send a string directly to the printer. This routine can be called from the user's program regard less of the operating mode of the trainer and DIP switch setting on the trainer.

"Print String" routine details are presented with a demostration example in chapter 10.

6.8 ESA 86/88E CONNECTORS

ESA 86/8E provides several connectors for external interfacing and system expansion. A brief summary of the connectors available on the trainer is described below. Refer the component layout diagram in Appendix C to locate these connectors. The signal definitions of these connectors are available in Appemdix B.

J1 : 5V Power Jack.

J2	:	9- Pin , D- type, female connector for RS-232-C standard.
J3	:	15- Pin Flow-Strip for LCD Module.
J4	:	26-Pin parllel I/O Connector (Default Provision.)
J5	:	26-Pin parllel I/O Connector (Optional).
J6	:	26-Pin connector for system expansion (Address and Control signals).
J7	:	26-Pin connector for system expansion (Data and Control signals)
J8	:	ASCII Keyboard Interface Connector.
J9	:	7-Pin Header for 8253 Timer/Counter signals.

CHAPTER 7

ESA 86/88E PROM PROGRAMMER SYSTEM

7.1 INTRODUTION

ESA 86/88E Monitor program supports all the function of ESA EPROM Programmer interface. Thus ESA 86/88E in conjuction with this interface module form a powerfull and easy to use EPROM Programmer system. This chapter describes the use of this EPROM Programmer system.

The system Permits the user to program, verify blank check & read any of the popula EPROMs - 2716 through 27512. The system hardware consists of the interface module connected to the parallel I/O Port of the trainer using a 26-core ribbon connector. the software is invoked from the ESA 86/88E monitor itself.

The EPROM Programmer interface module provide a 28-pin ZIP socket for placing the EPROMs When a 24-pin EPROM is to be places, it must be aligned with the bottom row i.e top two rows of zip socket are to be left blank. the system uses Intelligent Programmin Algorithm whenever possible thus reducing the programming time significantly. the devices supported by the system, their programming voltages and the type number to be entered by the user are listed below: the device selection is totally software-controlled. No hardware change are necessary for working with any of these devies.

Device (EPROM)	Programming Votage (Vpp)	Vpp Pin No. on zip Socket	Type Number to be entered
2716	25V	23	2716
2732/ 2732 D	25V	22	2732
2732 A	21V	22	732 A
2764	21V	1	2764
27C64/ 2764A	12.5V	1	764 A
27128	21V	1	0128
27128A /27C128	12.5V	1	128A
27256 / 27C256	12.5V	1	0256
27256	21V	1	2256
27512 / 27C512	12.5V	22	0512

NOTE:

1. ESA EPROM Programmer interface is optionally available with ESA 86/88E and is not part of the default scope of supply. Please contact the manufacturer for further details.

2. The description given in this chapter holds good for either mode of trainer operation.The output of the commands are described, as they would appear on the console in serial mode. while running the dos driver package XT863.EXE.The only difference that would occur in the stand-alone mode of operation would be in the output formatting on the LCD.

7.2 INSTALLATION

1. Turn off power to ESA 86/88E trainer.
2 Attach the hardware module to ESA 86/88E over the connector J4 using the 26-core ribbon cable supplied with the module.
3. Connect the black, yellow and blue wires coming from the 4-pin polaized connector on the Programmer module to the corresponding power supplies as shown below.

Color of the wire	Supply to be connected
BLACK	GND
YELLOW	+ 12V
BLUE	+ 30 V

4. Power ON the system.

CAUTION: The following precautions must be taken failing which, the device in use as well as the PROM Programmer system is liable to be damaged.

i. Spurious application of programming voltage to EPROM during switching operations with ESA 86/88E may damage the EPROM. To avoid this, insertion/ removal of EPROM should be done only when the trainer is powered ON.

ii. Ensure that the device type number entered corresponds to the EPROM inserted in ZIF socket.

iii. Avoid attempting to program already damaged devices.

iv. Avoid resetting the trainer while any of the above operation is in progress.

7.3 OPERATION

The EPROM programmer can be invoked either from the serial or the stand-alone monitor. The following discussion holds good for operation with the EPROM Programmer system for both modes of Progrmmer system for both modes of trainer operation except wherever indicated.
Enter P when prompted for command entry. Now, the EPROM programmer software gains control of the system and it will a display a list of EPROM Programmer commands as follows

P:Program R:Read B:Blank Check V:Verify E:Exit to Monitor
Select Option:

This is the command prompt of EPROM Programmer system for ESA 86/88E Trainer. Enter either of the following characters to proceed with the appropriate EPROM Programmer function (P/R/B/V).Enter 'E' to exit to the system monitor.

The parameters required for each of the above commands are described in the following sections For all these commads the system shows the default values. The user may modify these parameters to suit individual requirements. To abort any command the user can enter <EscL>; then the system will return to the EPROM Programmer command prompt.

Note: All commands of ESA 86/88E EPROM Programmer system except Blank Check use the trainer memory as buffer space. Thus it is advisable NOT to use the monitor's stack area (i.e. locations 0:0 to 0:1FFFH) for operation with this interface as it may lead to unpredicatable results.

7.3.1 P (PROGRAM) command

This command is used to program an EPROM. Enter P at the EPROM Programmer command prompt to execute this command, This command requires the following four parameters.

PROM Tyoe : EPROM Type (One of the type listed in section 7.1)

Buffer Start :

Segment : Segment address of the stating location of the buffer. The default. value will be the current value of CS register

Offset : Offset from the starting address of the current segment. the default value will be the current value of IP register.

Buffer End : Ending addresss of the buffer. Default value of this paremeter is the maximum buffer offset value calculated using the device size and the user specification if Odd, Even or All location from the buffer are being programmed.

PROM Start : Absolute starting address of the EPROM (from where programming its to begin)

As soon as 'P' is typed , the system will display the current values for the above parameters. the user may enter new parameter values value followed by **<CR>** or simply enter **<CR>** if the displayed value is not to be changed.
In case of any invaild entry, the system displays an error message **"Invalid Parameters"** for the parameter again. Entering **<Esc>** at any stage will cause the system to return to PROM Programmer prompt.

Note that the parameters must satisfy certain conditions as listed below.

i) PROM type should only be one of the type listed in section 7.1
ii) Buffer start offset and Buffer End address must either be either even or odd addresses.
iii) Buffer end address can be only an offset (Segment is not allowed). The segment value is the one specified for the buffer start address.
iv) Buffer end address must be greater than or equal to the offset value of the Buffer start address.

v) The EPROM must have enough space to accommodate all the bytes specified by the Buffer start address and Buffer end address i.e. the following relation must be satisfied.

EPROM Start + [(buffer end address- Buffer Start address) 2] <= Highest absolute address of the EPROM.

NOTE : 8086-memory space consists of an even bank and an odd bank. Thus an EPROM is programmed from alternate source bytes i.e. only bytes form successive even or odd address location used. Although 8088-memory space is linear, it is organized as even and odd arrays on ESA 86/88E.

Example : For EPROM type 2764, the highest absolute address is 1FFFH. Suppose the parameters entered are:

PROM Type	:	2764
Buffer Start	:	
Segment	:	
Offset	:	2000
Buffer end	:	5FFE
PROM Start	:	100

Then 100 + (5FFE-2000)/2= 20FF>IFFF so this combination of parameters is invalid.Hence the error message **"Invalid Parameters"** is displayed.

After correct entry of the paramete values by the user, the system checks the EPROM for blank value (OFFH) for the specified address range. During this time, the message **Blank checking...**is displayed.

NOTE: The system will check whether the EPROM is blank only in the range specified by the command parameter. Thus the user can check if only a particular block of address in the EPROM are blank, regardless of whether the EPROM contains data elsewhere.
If the EPROM is not blank, the following appears:

PROM not blank !!!
Non-blank at -XXXX

Where XXXX is the address of the first non-blank value found in the device.

Then, the system returns to the EPROM Programmer command prompt.

If the EPROM is already blank, following message appears:

PROM is blank....
Blank check completed

Then the system proceeds with programming the device with the data specified by the buffer limits and will display

Programming...

The system also verifies the programmed data on a byte-by-byte basis. intelligent programming Algorithm is used if the EPROM can support it. This results in considerable reduction in programming time.

If complete programming is successfull, the system will display the following message.

Progrmming Completed
Check Sum...XXXX.

Where XXXX is a 16-bit checksum and control will return to the EPROM programmer command fetch routine.

If progrmming is unsuccessful, the following information is displayed.

Failed At - XXXX

Where XXXX indicates the location in HEX where programming was unsuccessful. the system then returns to EPROM Programmer command prompt.

EXAMPLE: Enter P at the ESA 86/88E Monitor Command prompt

.P<CR>

P:Program R:Read B:Blank check V: Verify E: exit to Monitor
Select Option : P<CR>

PROM Type : 2716-2764<CR>
Buffer Start :
Segment : 0000<CR>
Offset : 0000-2000<CR>
Buffer End : 5FFE-2FFE<CR>
Invalid Paraments

This is because the buffer end address is an old location, while the buffer start offset address is an even location i.e. the even bank of memory is selected for use.

Buffer end : 5FFE-2FFE<CR>

PROM Start : 0000-2000<CR>
Invalid Parameters

This error occurred because the EPROM does not have enough location to be programmed as that indicated by the buffer range.

PROM Start : 0000-1800<CR>

Blank Checking...

PROM is blank...
Blank Check completed

Programming...

Programming Commpleted
Check Sun ...0FC2

Now the programming is complete and the system returns to EPROM Progrmmer command prompt

7.3.2 R(READ) Command

This command is used to transfer the contents of the EPROM into the ESA 86/88E memory space. Enter **'R'** to execute this command. This command requires the following parameters.

PROM Type : EPROM Type (one of the types listed in section 7.1)

PROM Start : Absolute starting address of the EPROM (from where reading is to begin.)

PROM End : Absolute ending address of the EPROM. (till which address to be read.)

Buffer Start : Starting address in ESA 86/88E memory space
This value can be an offset only; a default and fixed Segment value=000H is used. The default Offset is set at 2000H, such that the Buffer Start value will point to the user RAM area. The offset value can however be modified by the user.

In case of any invalid entry, the system displays an error message **"Invalid Parameters"** and prompts for the parameter again.

The parameters must satisfy the following relations.

i) PROM Start <= Absolute end address of the EPROM
ii) PROM End <= Absolute end address of the EPROM and >= PROM Start.

if all parameters are entered correctly, the system proceeds with reading the device and displays

Reading...

The EPROM data is transferred into ESA 86/88E memory, starting from the location specifed as buffer Start address into successive odd or even locations. Once all the bytes specified in the EPROM range are read the system displays a 16-bit Check Sum of the data read and indicates whether reading is completed.

Check sum....XXXX
Reading complete.

If however, all the bytes are not transferred into ESA 86/88E memory space, then the system displays the number of bytes read from the EPROM (In Hex) and a 16-bit check sum of these bytes (NNNN) as shown below. Such a situation is encountered when the buffer range specified is lesser than the EPROM range. Note that in this case the message **'Reading Completed**.' is not displayed.

NNNN Bytes Read
Cheak Sum...XXXX

The system then returns to the EPROM Programmer command prompt.

Note : Selecting the ESA 86/88E Monitor stack area i.e. location 0:0 to 0:1FFFH the buffer may lead to unpredictable results.

7.3.3 B (BLANK CHECK) Command

This command is used to check if a specified range in the EPROM is blank (contains 0FFH). Enter **'B'** to execute this command. This command requires the following parameters.

PROM type : EPROM Type(One of the types listed in section 7.1)

PROM Start : Starting address of the EPROM, from where blank checking is to begin.

PROM End : Ending address of the EPROM, till which Blank checking is to be done.

In case of any invalid entry, the system displays are error message **"Invalid Parameters"** and prompt for the parameter again. The parameters must satisfy the following relations.

i) PROM Start <= Absolute end address of the EPROM
ii) PROM End <= Absolute end address of the EPROM and >= PROM Start.

NOTE : The system will check whether the EPROM is blank only in the range specified by the command parameters. Thus the user can check if only a particular block of address in the EPROM are blank, regardless of whether the EPROM contain data elsewhere.

If all the parameters entered are correct, the system proceeds with reading the data from the device and comparing it with the device blank state value (FFh in case of EPROMs) During this times the following message is displayed.

Blank Checking.....

If all the location are blank then the following message is displayed, and control returns to EPROM Programmer command prompt.

PROM is blank...
Blank Check completed

If any memory location in the specified range is not blank, the corresponding message and the absolute EPROM address of the first non-blank location is displayed as shown below.

PROM not blank !!!
Non blank at - XXXX

Where XXXX is the address of the first non-blank value found in the device. Note that only the first non-blank location address is displayed. Subsequent location may or may not be blank.

7.3.4 V (VERIFY) command

This command is used to verify the contents of an EPROM agains a source. Enter **'V'** to execute this command. The parameter and their interpretation are completely similar to that with respect to **'P'** command.

PROM Type : **EPROM type (one of the type listed in section 7.1)**

Buffer Start :

Segment : Segment address of the starting location of the buffer. The default value will be the current value of CS register.

Offset : Offset from the starting address of the current segment. the default value will be the current value of IP register.

Buffer end : Ending address of the buffer, Default value of this parameter is calculated similarly as done with the 'P' command (explained in Section 7.3.1)

PROM Start : Absolute starting address of the EPROM, from where verification of the device should begin)

In case of any invalid entry, the system display an error message **"Invalid Parameters**" and prompts for the parameter again. When all valid parameters are entered the following message is displayed on the console.

Verifying....

If the verification is successful, the appropriate message along with a 16-bit checksum of the data verified is displayed as follows.

Verify Completed.
Check sum...XXXX

If the verification fails, a corresponding message is displayed as follows, indicating the first mismatch of data between the buffer and the EPROM.

Verifly Failed At - XXXX
PROM Data - XX
Buffer Data - XX

Where XXXX and XX are address and data in Hex format respectively, The system then returns to the EPROM Programmer command prompt.

7.3.5 E (EXIT) command

This command is used to terminate the EPROM Programming software and return control to the ESA 86/88E monitor. The monitor sign-on message then appears on the console or the LCD followed by the command prompt on the next line.

CHAPTER 8

ESA 86/88E SYMBOLIC ONE-LINE ASSEMBLER

ESA 86/88E provides a very convenient environmentfor Assembley Leve Programming in either mode for trainer operation. This chapter describes the use of the powerfull monitor resident ESA 86/88E Symbolic One-Line Assembler along with the other programming facilities viz. Label Commands and disassembly Appendix F contains a list of 8086/8088 CPU. instrutions supported by the assembler. Note that this assembler does not support 8087 NDP instructions.

8.1 ESA 86/88E SYMBOLIC ONE-LINE ASSEMBLER

The monitor resident Symbolic One-line assembler provided with ESA 86/88E Trainer is capable of translating the mnemonic Instruction code to equivalent machine codes for the 8086/8088 CPU. The Translated code is immaediately loaded into appropriate memory locations. The assembler supports the standard 8086/8088 mnemonic syntax with some simple and easy to understand modifications. This assembler supports labels (symbolic references) also, that can have a maximun length of three significant characters. However a label can be referenced only if it has already been defined i.e. the assembler supports backward references only .

In addition to the standard instruction mnemonics, ESA 86/88E symbolic One-Line Assembler Supports some useful Assembler directives (Pesudo op-codes). These directives can be used to set the origin, define symbolic constant, initialize byte, word and string values, and to reserve memory space.

8.2 ASSEMBLY COMMADN(A)

8.2.1 FORMAT AND OPERATION

The command syntax for invoking ESA 86/88E Symbolic one-line assembler is

A[<address>]<CR>

The Assembly command has a single optional command parameter, which is the assembly address for source instructions. when prompted for command entry, enter 'A' followed by the assembly address and **<CR>**, Now the monitor will transfer control to the Assembler and the user can enter any of the 8086/8088 CPU instructions or directives supported by this Assembler. These commands and directives are described in detail in later sections.

NOTES

1. The assembly address is optional and if entered, the assembler will start assembling source statements from this address. In case , no address is entered the assembler will use the current content of the CS and IP registers as the effective address for assembly.

2. While using the Assembler, the user must enter source statements/programs only in the User RAM area starting from location 2000H. Entering program in reserved location may lead to unpredictable results.

3. Other ESA 86/88E monitor commands cannot be executed while the assembler in active.

8.2.2 ASSEMBLER SYNTAX DESCRIPTION

Once the assembler is involed the system will output the assembly address in Segment:Offset format followed by the instruction code and disassembled instruction at this address. On the next line, the assembler will display the assembley prompts **A:** and wait for the user to enter a source statement.

Now user can either enter a source statement, carriage return if the current instruction is not to be changed or the character '!' and <CR> to terminate the command.

If the user enters a new source statement, it is immediately translated and the machine code generated is stored at the appropriate memory locations. If any errors are detected or if the statement is written at an invalid location (viz. EPROM Area or non-existent memory), appropriate error message is issued next line and location counter is not updated and the old line is displayed again. If there are no errors, the assembled machine code is immediately loaded into the memory; the valid source statement and its machine code will overwrite the previous mnemonic and its corresponding machine code while updating the location counter. Now the next location counter (address) is displayed followed by the instruction code and disassembled instruction at this address. This is followed by the assembly prompt **A :** on the next line

NOTES

- The source line entered by the user is analyzed only after a carriage return is entered. Thus user can carrect the entry errors using the Backspace key.
- The address displayed when the assembler is invoked will bear the current CS and IP register contents unless the user specifies a different address with the 'A' command.
- If the user doesn't wish to change the displayed instruction he can enter the carriage return.
- The location counter will be updated; the new value of the location counter is displayed along with the machine code, the disassembled instruction at that location and the assembly prompt on the next line. The process continues thereafter.
- If the use enter '!' followed by <CR> the assembler is terminated and control returns to the ESA 86/88E Monitor command prompt.

EXAMPLES

1. Invoking the ESA 86/88E symbolic One-line Asssembler at the current PC value.

```
.A<CR>
0000:2000     FE CO          INC    AL
A:
```

Here, the CS & IP register contents before invoking the Assembler are assumed to be 000H & 2000H respectively. The display format shown is for the serial mode of operation. The location counter is followed by a string of the machine code and the current instruction at that address. Further, the system will output the Assembly prompt **A:** in the next line.

In the stand-alone mode the display format on the LCD is slightly different and is as given below. In this format, the display string following the location counter is the current instruction at that address. This is followed by the machine code for that instruction and the assembly prompt **A :** on the next line.

```
.A<CR>
0000:2000
INC    AL
FE     C0
A:
```

2. Invoking the ESA 86/88E Symbolic One-line Assembler with a specific address.

```
.A  0:2500 <CR>
0000:2500 8B   C3     MOV   AX,BX
A:
```

The output on the LCD in case of stand-alone mode of operation will be as follows

```
.A  0:2500 <CR>
0000:2500
MOV   AX,BX
8B C3
A:
```

3. Writing a small program using ESA 86/88E Symbolic One-line Assembler. This example illustrates the use of the assembler in Serial Mode to write a small program. Some common errors are also described in the example and explanation pertaining to these errors and display format is given at the end of the example.

```
.A<CR>
0000 : 0000 00 00              AND          [BX] [SI], AI
A:      ORG 2000<CR>
0000 : 2000 E0 E9              LOOPNE       1FEB
A:      MOV AX,AABB<CR>
Invaild Operands.
0000:2000 E0 E9                LOOPNE       1FEB
A:      MOV AX,AABB <CR>
        -> B8   BB  AA
0000:2003      BO   90         MOV          AL, 90
A :     MOV BX, 2222 <CR>
        -> B8   22   22
```

```
0000:2006    03     C3     ADD   AX,BX
A:      <CR>
0000:2008    50     PUSH          AX
A:     1VAL: DB     10
Lables should start with Alphabet.
0000:2008    50     PUSH          AX
A:     VALUE: DB    10<CR>
       ->DB  10
0000:2009    FB     STI
A:     MOV  CL, @VAL     <CR>
       -> 8A  OE     06     20
0000:200D    26     ES
A:     UP:   INC    BX<CR>
Invalid Mnemonic.
0000:200D    26     ES
A:     UP :  INC    BX<CR>
       ->43
0000:200E  C3        RET
A :    INC   [SI] <CR>
Illegal Operands.
0000:200E    C3     RET
A:     INCW  [SI[<CR>
       ->FF  04
0000:2010    34     01     XCHG AL,01
A:     JMP   UP<CR>
       ->E2  FB
0000:2012    CC     INT    03
A:     <CR>
0000:2011    EE     OUT    AL,DX
A:     !
.
```

NOTES :

1. The Assembler is invoked without any specific address. Hence it takes the current CS & IP register contents as reference memory locations.

2 Use of Origin Control Directive (ORG) is made here before entering any instructions since the current location counter does not point to user RAM area. The location counter is now modified to point to the address specified by this directive.

3. Error message follows the first instruction entered here, since the Hex operand does not begin with a numeric. The location counter is not updated until a valid source statement is entered. In such case, the appropriate error message is displayed and the current location counter is displayed again followed by the machine code and instruction at that address and the assembly prompt A: on the next line. The Mnemonics Syntax for ESA 86/88E Symbolic one-line Assembler is described in detail in a later Section.

4. When an instruction is successfully assembled, the machine code for the instruction is output on the next line as shown in the third instruction entered in the above example. this is preceded with a -> symbol and the location counter is suitably updated and subsequently displaye with the assemblly prompt.

5. If the user does not wish to change the current instruction at a location, then entering a **<CR>** at

   that location will maintain the instruction and update the location counter, Note that in this case, The current line is overwritten by the next location lines in the above example indicate that, they are overwritten by the next output line. This is illustrated in the instruction at location 0:2006H and 0:2012H

6. Lable supported by the assembler can have a maximum of three significant characters, if the lable contains, more than three characters, then they can be referred using only the first three characters. The ESA 86/88E Monitor supports some additional label commands that are discussed later. Also refer section 8.3.5 for more details on the usage of lables.

7. Almost all instructions mnemonics may be 'optionally' suffixed with the letter **'W'** or **'B'** to indicate word or byte operation. **However for indexed addressing mode instructions that have only one operand pointing to the effective address,this suffix becomes mandatoryfor the correct assembly of the instruction.** This is equivalent to specifying **Word Pointer** or **byte pointer** with the mnemonic, as is the case with some other asembler like MASM, TASM, 2500AD or Micrsoft Debug.

Example: for the instruction **INC [BX] [SI] 1234,** the source statement to be entered during assembly will be **INCW [BX] [SI] 1234** for word operation. If byte operation is intended, then the source statement will be INCB **[BX] [SI] 1234.**

8. The user can terminate the active assembler by entering **'! <CR>'** at the Assembly prompt A:. The user will now be to use other ESA 86/88E Monitor commands.

Using ESA 86/88E Symbolic One-line Assembler in Stand-alone Mode of Operation:

In stand-alone mode of operation, the display format on the LCD will follow the same lines as that described in Example 1&2. However some points must be noted here.

- The Assembly prompt follows the location counter, the instruction at that address and the machine code in subsequent lines and generally occurs on the fourth line of the LCD.

- When an instruction is successfull assembled,the display is refreshed and the next location counter with the corresponding instruction and machine code is directly displayed. Thus the user may understand that a source statement has been successfully translated by watching the location counter.

- In case of an error, the error message is displayed after refreshing the display once.Now, the assembler waits for a user strobe viz. <CR> and then display the same location counter with the respective parameters as explained before.Thus, the display pattern while entering the second source statement in the above example in stand-alone mode will be as described below.

0000:2000
LOOPNE 1FEB
E0 E9
A: MOV AX,AABB<CR>

The LCD is now refreshed once and will display the error message

Invalid Operands.

Now, the Assembler waits for a **<CR>** from the user to proceed with the assembly proceed with the assembly process, When the user enter **<CR>** the display is refreshed again and the same location counter with the corresponding parameters is output as follows.

0000:2003
LOOPNE 1FFB,
E0 E9
A: MOV AX, 0AABB<CR>

Upon successful assembly, the next location counter is directly displayed as shown below.

0.000 : 2003 ;
MOV AL, 90
B0 90
A :

- If the user wishes to view the machine code generated for the instruction entered, then he can do so by using the Origin Control Directive (ORG) specifying the previous address in the statement.

For instance, the user can view the machine code generated in the previous example by following the pattern described below.

0000:2000
LOOPNE 1FEB
E0 E9
A: MOV AX, 0AABB<CR>

0000:2003
MOV AL,90
B0 90
A: ORG 2000<CR>

0000:2000
MOV AX,AABB
B8 BB AA

A :

It may be seen from the above that ESA 86/88E provides an easy-to-use assembly level-program ming environment even in the stand-alone mode The user is urged to try out the above example in both mode of trainer operation to get well acquainted with ESA 86/88E Symbolic One-line Assembler. of operation Other assembly support instructions viz. label commands and disassembly, are discussed in subsequent sections.

NOTE : Before invoking ESA 86/88 E Symbolic One-line Assembler for the first session it is advisable to use label clear command *(described later),* to clear all random label definitions in the symbol table.

8.3 ASSEMBLY LANGUAGE CONVENTIONS

The different fields in a source statements is described in this section Each line of a source statement can contain a lable field, a mnemonic field & an Operand field in that order. Thus the general syntax for a source statement involving a CPU instruction will be

[label]< Mnemonic>[operand(s)]

The different fields in a source line are identified by the order in they appear. These fields are separated by one or more blank spaces or a comma acting as the delimiters.

8.3.1 LABEL FIELD

The label field is not exclusive in nature and may be overwitten by mnemonics or assembler directives that are part of the source statement. A label can have a maximum of three significant characters. The first label character must be an alphabetic character. The following character if present can be an alphanumeric . if more than three character. The following characters if present can be an alphanumeric If more three characters are assigned to the label, the assembler stores only the first three characters and ignores the others. Register names and assembler directives given in the list below form reserved word and cannot be used as labels.

AH, AL, ASC, AX, CS, DB, DS, DSP, DW, EQU, ORG, SS, ES CH, CL ,CX,BH,BL,BX, SP,BP, SI, DI,DH,DL,DX.

8.3.2 MNEMONIC FIELD

The next field is the mnemonic field. This field can overwrite the label field and always begins with an alphabetic character. This field is also used for entering Assembler Directives as source statements.The assembler supports the standard INTEL mnemonics for 8086/8088 CPU instructions, with some easy to understand modifications in a few cases.

The detailed syntax of mnemonics and addressing mode supported by ESA 86/88E Symbolic One-Line Assembler is given at the end of this chapter.

8.3.3 OPERAND FIELD

The operand field follows the label and mnemonic fields. This field holds the operands specific to the instruction. An instruction may or may not have an operand at all. The operands can be register symbols. data addresses. or labels. This assembler supports all standard addressing modes of 8086/ 8088 CPU. Please refer to the Mnemonic syntax description, the disussion on labels and programming example given in the manual to get familiar with the mnemonic and operand syntax supported by this Assembler.

Labels can be used in the operand field, provided that they are been already defined.

8.3.4 SEGMENT OVERRIDE OPTION

Segment override option can be exercised by giving the segment register instruction prior to the actual instruction e.g. "MOV ES:[S1], AL" instruction specifies a byte movement with the segment register ES overriding the default segment register DS. To realize this instrution in the present assembler, the user must enter the following source statements.

```
ES      <CR>
MOV  [SI] ,AL
```

NOTE: The segment register override instruction will be applicable for all the instructions that follow it. The assembler will not check for the override capability of the next instruction, Thus if you want the overrid only one instruction, you must restore the default segment register explicity.

8.3.5 MORE ON USAGE OF LABELS

Whenever labels are included in source statements, the assembler understands them as address references. The following discussion on labels should serve as guideline during program assembly using ESA86/88E Symbolic one-line Assembler.

- The label definitions can have a maximum of three significant characters.
- If a previously defined label is defined again during assembly, then the new definition is also recorded in the symbol table. All future references to this label will result in only the new address being referred.
- More than one label can be defined at one location. In such cases, using any of these labels can refer this location.
- Labels can be used with Assembler directives also.However, if used with ORG directive, there is no lable assignment in the symbol table since this directive does not perform any memory allocation.
- Label referencing for different addressing modes can be done using the label name directly with the instruction.

EXAMPLE (Refer Section 8.5 also)

1. The instrution **MOV AX,R1** Will move the address location referenced by R1.if R1 is defined at 0:2500H,then executing this instruction will transfer 2500H to AX register.

2. The instruction ***MOV AX, @R1*** will move the contents at the location reference by R1.
3. The source statement ORG R1 will direct the assembler to begin assembly from location referenced by R1
4. The same holds true for branching instructions viz LOOP and JUMP.

8.4 LABEL COMMANDS

ESA 86/88E Monitor provides two support commands for the assembler pertaining to usage of labels. The commands are briefly discussed here.

1. LABEL CLEAR (LC)

This command clears all labels from the symbol table in the system memory.

FORMAT & OPERATION

LC<CR>

To use this command in either mode of operation, the user has to enter **LC** followed by a <CR>.

This command will clear the symbol table and all label definitions generated in the earlier invocations of the assembler are lost.If this command is not used, earlier label definitions will remain valid for this session and until the command is used.

It is urged that this function be used whenever a fresh session with the trainer is begun especially before invoking the assembler for the first time. This function will clear all random lable assignments.

2. LIST LABELS (LL)

This command output the total count of label definitions in the symbol table and lists them.

FORMAT & OPERATION

LL<CR>

To use this command in either mode of operation,enter **LL** followed by a <CR>.

This command first output the Label Count, which is a hexadecimal count of the number of labels defined in the symbol table. The symbol table follows the FIRST IN FIRST OUT method of storing label definitions and hence, the order of label list is independent of address sequence. The labels are listed is independent of address sequence. The labels are listed in the order the they are defined.

If there are no labels defined in the symbol table, this command will output a label count of zero with the message ***'No Labels Defined.'*** Thus this command can be used to check the validity of the Label Clear command also.

If the same label name or symbol is defined more than once the command will output all the symbols with their addressess.

EXAMPLES

1. User of list labels command when no labels are defined.

```
.LL<CR>
Label Count ; 0000
No Labels Defined.
.
```

This example show the output for the LL.Command in the serial mode when there are no labels defined in the symbol table. In the stand-alone mode of operation, the monitor refreshes the LCD before displaying the label count and the message.

2. Use of list labels command in stand-alone mode of operation.

```
.LL<CR>
```

(The monitor first refreshes the LCD and then displays the label count).

```
. Label Count :0006
```

(The monitor now waits for a user strobe viz.<CR> before listing the labels defined).

```
<CR>
0000:2000     R1
0000:2005     BAC
0000:200A     L1
0000:2002     R2
<CR>
0000:3000     L2
0000:3000     UP
.
```

This example ilustrated the use of LL command in stand-alone mode of operation. The listing format includes the address location of the label of the label followed by the symbol (LabelName) defined. This format is the same for output in serial mode also.

Note that after listing four consecutive label definitions, the monitor waits for a user strobe viz <CR> before proceeding with the output. In the serial mode the monitor waits after listing 25 label definitions and then displays a message **'Press Any Key'** The listing continues upon pressing any key. In either mode of operation, entering <ESC> Key while the monitor is waiting terminates the command and the monitor returns to command prompt mode.

8.5 ASSEMBLER DIRECTIVES (PSEUDO OP-CODES)

In addition to the normal op-codes that generate executable machine instructions the Assembler recognizes some pseudo op-codes which ocupy the mnemonic field like normal op-codes. These directives instruct the assembler to perform certain functions like setting the origin, defining symbolic constants, initializing byte, word or string values, etc. ESA 86/88E symbolic one-line Assembler supports the following assembler directives.

- Origin control (ORG)
- Symbolic constant definition (EQU)
- Byte initialization (DB)
- Word initialization(DW)
- String constant initialization(ASC)

Each of these directive is now discussed briefly.

1 Origin control (ORG)

ORG <Value><CR>

The location counter or address can be set to a specific value with the help of this directive.

It is generally used at the first program entry for the starting location of the assembled code. For example **ORG 2000** will result in the loading of next instruction assembled to start from location 2000H.The value to be specified with this directive can be entered in the **Segment:Offset format** also.

Example:

```
.A<CR>
0000:0000    27              DAA
A:     ORG   2000<CR>
0000:2000    00    3F              ADD   [BX],BH
A:     ORG   0F00:  0<CR>
F000:0000    FA              CLI
A:
```

This example illustrates the use of ORG directive to set location counters. After invoking the assembler the location counter corresponds to the current content of the CS and IP registers, The first ORG Directive is made to change the offset of the location counter only and hence no segment specification is made In the second source statement ORG is used with segment as well as offset specification. This statement modifies both the segment and the offset of the location counter accordingly.

2. Symbolic Constant definition (EQU)

User can define a value for a symbolic constant by using the 'equate' (EQU) directive.

<Symbol> EQU <defined constant><CR>

For example, the symbol T1 can be defined to be equal to 2000H, by entering **T1 EQU 2000**

Further the EQU directive allows the user to assign the value of another symbol. for example, if T1 is already defind as 2000H, then entering T2 EQU T1 defines T2 also as 2000 H

```
T1      EQU   2000        : T1 represents 2000H
T2      EQU   T1          : T2 represents 2000H
Y       EQU   10          : Y represents 10H
```

Where using the symbols defined by this directive, the symbol name can be used directly with the instruction if the value to which it is assigned is required. If however a reference to the location pointed by the symbol is required, then it can be done by using the character '@' before the symbol name.
Exmaple : With the symbols is defined above, the instruction MOV BX, T1 will transfer the data word 2000H to BX register while the instruction MOVW BX, @T1 will move the data word indexed by CS: 2000H to BX register.

3. Byte Initialization(DB)

[<label>] DB<value>[,<value>]<CR>

The user can initialze a memory location to a particular value by the use of this command . This facility is particularly useful when entering a table of data as part of a program. This directive supports up to two constants separated by a comma or space.

Eg:

```
0000:2240    44     INC   SP
A:     T1    DB     1240<CR>
->DB  12,0A4
```

This source statement will initialize location 0:2240 with the value 12H and location 0:2241 with the value A4H, Further the symbol T1 is made equal to 2240H.

4. Word Initialization (DW)

[<label>] DW <value> <CR>

This directive is similar to DB directive except that with this directive, a word of memory rather than a byte is initialized. Further, only one value is permitted with one statement.

E.g. :

```
0000:2240    44     INC   SP
A:     TY    DW     1240<CR>
->DW  1240
```

will initialize the word location at 0:2240H with the value 1240H.

5. String constant Initialization (ASC) ASC ' String'

This directive allows the user to enter a string of characters and have these characters translated to ASCII Codes and stored in the memory. The string can have a maximum of 160 characters. The string must be enclosed within a pair of single quote (') characters.

E.g.

```
0000:2000    90     XCHG AX,AX
```

A : ASC 'ESA' <CR>
- > ASC 'ESA'

This source statement will initialize locations 0:2000H, 2001H & 2002H respectively with 45H, 53H and 41H. Now the assembler will overwrite the original line, with the string displayed in op-code field.

8.6 DISASSEMBLY COMMAND (Z)

ESA 86/88E Moniter provides an extremely useful command viz. Disassembly in support of the Symbolic One-Line Assembler. With the help of this command the user can disassemble a range of memory locations into corresponding Assembly Mnemonics. Thus the user can view the program and subsequently verify it even if machine code has been entered directly.

FORMAT

Z[<start address>[,<end address>]]<CR>

OPERATION

- The command syntax for Disassemble command is similar for either mode of trainer operation.
- To use this function, enter **Z** when prompted for command entry. The user may now enter a <CR> to view the disassembled instruction corresponding to the current CS and IP register contents.

If the user wishes to view the instruction code at only a particular memory location, then that address must be entered followed by a <CR>.

To view the disassembled instruction Mnemonics for a range of locations, the user must specify the same by entering the start and end addresses of the range separated by a comma. Upon entering <CR> now, the monitor will output the disassembled instructions in the following format.

In Serial Mode, the following pattern will appear on the console

Segment:Offset Machine Code Mnemonic

In stand-alone mode operation the same parameters will appear on the LCD in the following pattern.

Segment : Offset
Mnemonic
Machine Code

After displaying the disassembled code, the monitor returns to command entry mode.

- The end address specified with the command is always an offset relative to the segment address with the start address. Hence a segment address should not be specified with the end address. However if no segment address is specified with the start address parameter, then the address range of disassembly is relative to current CS register contents.
- In stand- alone mode, the monitor waits for a user strobe after displaying one location, the instruction and the machine code at that location and then either returns to command prompt or refreshes the LCD and displays the next disassembled location, the instruction and the respective code. In serial mode, the entire specified memory range is disassembled and displayed as a continuous list after which the monitor return to command prompt.
- In serial mode operation, display control commands viz.**<Ctrl-S>, <Ctrl-Q>** and **<Ctrl-S>** can be used to control the output flow to the console. (Refer Section 4.4)
- The labels or symbols defined during assembly are not displayed during the disassembly; appropriate address references or values defined using symbolic constants replace the labels and symbols defined.

EXAMPLES

1 Disassembling the Hex code at location 0000:2000H. Assume CS content is previously set to 0000 in serial mode of operation.

```
.Z      2000<CR>
0000:2000    92            XCHG AX,DX
.
```

2. Disassembling a range of memory location from 0200:0030H to 0200:0040H.

```
.Z 200:30 , 40<CR>
0200:0030    BA    E7    FF    MOV  DX,FFE7
0200:0033    F7    DO          NOT  AX
0200:0035    92                XCHG AX ,   DX
0200:0036    B9    01    00    MOV  CX,    0001
0200:0039    D2    FB          SAR  BL,CL
0200:003B    9A    00  10  00  FB CALLS FB00:1000
0200:0040    CC                INT  03
.
```

In stand-alone mode of operation, the output for the same command will be in the pattern described below, As mentioned earlier, the monitor waits for a user strobe after diassembling and displaying one location and its corresponding parameters.

```
.Z 200:30,40<CR>

0200:0030
MOV  DX,FFE7
BA    E7    FF<CR>

0200:0033
NOT  AX
F7 DO <CR>
```

0200:0035
XCHG AX,DX
92<CR>

0200:0036
MOV CX,0001
B9 01 00
<CR>
0200:0039
SAR BL,CL
D2 FB
<CR>

0200:003B
CALLS FB00:1000
9A 00 10 00 FB
<CR>

0200:0040
INT 03
CC
<CR>
.

- Since the end address offset specified with the command is relative to the segment value of the start address, this command can disassemble and display a maximum of 64K of machine code in a single operation.
- Specifying an end address offset value lesser than the start address offset results in an error.

8.7 ADDERSSING MODES

ESA 86/88E Symbolic One-Line assembler supports all the addressing mode of 8086/8088. the syntax to be followed for the various addressing modes is summarized as follows Please refer Appendix F for the list of Instruction mnemonics and the syntax supported by the assembler.

Base Register [BX]or [BP]
Index Register [SI] or [DI]
Base+Displacement [BX],[SI],[DI] or [BP] followed by value or label.
Base+Index [BX][SI],[BX][DI],[BP][SI] or[BP][DI]
Immediate data VALUE or LABEL
Direct addressing [SI],[DI],@ VALUE or @LABEL
Displacement+Base+Index [BX][SI] VALUE or[BX][SI] LABEL

NOTE

1. All numerical values are at hexadecimal base.
2. Label when used, must have defined in previous instructions.
3. Mnemonics may be appended with a 'B' or 'W' to specify whether the operands are 'bytes' or 'words' respectively.

CHAPTER 9

ESA 86/88E MONITOR ROUTINES

ESA 86/88 E Monitor provides the user with several useful routines in either mode of trainer operation that significantly simplifies the task of program development. This chapter gives a list of routines provided by ESA 86/88E Monitor, which are accessible to the use with their address. Example programs using these routines are given in Chapter 10.

The outputs of some routines are independent of the mode of trainer operation while some routines provide input and output facilities only the default system I/O for a particular mode of operation.

Routines listed in section 9.1 are dependent on the mode of trainer operation. When used in stand-alone mode, the output will be displayed on the LCD, and in case of serial mode of operation the corresponding outputs will appear on the console. Similarly any input parameters that are required should be entered by the terminal keyboard in case of serial mode and by the PC keyboard interfaced with the trainer during stand-alone mode operation.

Routines listed in section 9.2 are independent of the mode of trainer operation and are useful in applications where independent control of the I/O devices is required.

9.1 MONITOR ROUTINES DEPENDENT ON OPERATING MODE

1. Name of routine: **OUT_CHAR**
   Function: Outputs an ASCII stored in AL register character to the console in Serial mode or to LCD in stand-alone mode
   Calling address: **FE00:0000**

2. Name of routine: **OUT_STRING**
   Function: Outputs a string of ASCII characters stored in memory. The String is indexed by ES:AX Where ES is the segment value and AX is the offset address of the starting location of the string.
   Calling address: **FE00:0013**

3. Name of routine: **OUT_CRLF**
   Function: Output a carriage return and line feed (ASCII code = 0AH, 0DH) to the LCD or console.
   Calling address: **FE00:0031**

4. Name of routine: **OUT_BLANK**
   Function: Output a blank character (ASCII code = 20H) to the LCD or console.
   Calling address: **FE00:0049**

5. Name of routine: **OUT_BYTE**
   Function: Outputs a byte in AL register to the LCD or to console
   Calling address: **FE00:0052**

6. Name of routine: **OUT_WORD**
   Function: Outputs a word value in store in AX register to the LCD or to the console.
   Calling address: **FE00:006A**

7. Name of routine: **OUT_ADDR**
   Function: Output an address value in **Segment : Offset** format to the LCD or to the console. The segment value to be displayed is stored in ES register and the offset in BX register.
   Calling address: **FE00:0082**

8. Name of routine: **GET_CHAR**
   Function: Reads an ASCII character from the keyboard and stores its Hex quivalent in AL register.
   Calling address: **FE00:00A9**

9. Name of routine: **GET_CHAR_E**
   Function: Reads an ASCII character from the keyboard and display the character on the LCD or console depending on the operating mode. the Hex equivalent is stored in AL register.
   Calling address: **FE00:00B8**

10. Name of routine: **GET_BYTE**
    Function: Read a byte value from the keyboard and stores it in AL register.
    Calling address: **FE00:00C7**

11. Name of routine: **GET_WORD**
    Function: Reads a word value entered from the keyboard and stores it in AX register
    Calling address: **FE00:00E0**

12. Name of routine: **GET_ADDR**
    Function: Read an address value from the keyboard in **Segment:Offset** format. The segment value entered is stored in ES register while the offset is stored in BX register.If no segment value is entered the current CS register value will be stored in ES registe.
    Calling address: **FE00:00F3**

13. Name of routine: **GET_STRING**
    Function: Reads a string of ASCII character from the keyboard and stores their HEX equivalent in memory indexed by DS:SI. The current content of DS register is the segment value and that of SI register is the offset value of the starting location of the string being stored. The string is terminated by <CR>.
    Calling address: **FE00:010E**

14. Name of routine: **VALID _HEX**
Function: This routine checks if the value stored in AH register is a valid ASCII Hex value (30-39 /41-46). Stores FFH in AL register if found vaild, and 00 if found invalid.
Calling address: **FE00:0123**

15. Name of routine: **HEX_ASCII**
Function: This routine converts a vaild ASCII Hex value (30-39/41-46) stored in AH register to its equivalent character (0-9/ A-F) and stores the result in AL register. Note that the routine will not convert any other values and that the previous contents of AL register are destroyed.
Example: 41H will be converted to 0AH : similarly 33H will be converted to 03H.
Calling address: **FE00:0131**

16. Name of routine: **ASCII_HEX**
Function: This routline converts a valid Hex character (0-9/ A-F) stored in AH register to its equivalent ASCII Hex value (30-39 / 41-46) and stores the result in AL register. Note that the routine will not convert any other values and that the previous contents of AL register are destroyed.
Example: 0AH will be converted to 41H: similarly 03H will be converted to 33H
Calling address: **FE00:0152**

17. Name of routine: **BCD_BIN**
Function: This routine converts a valid BCD value (up to 99) stored in AH register to its equivalent binary value in AL register.Note that the previous contents of AL register are destroyed.
Example: 99 will be converted to 63; similarly 10 will be converted to 0A.
Calling address: **FE00:016B**

18. Name of routine : **BIN_BCD**
Function: This routine unpacks a 8-bit binary number to three unpacked BCD digits and stores them in 3 consecutive locations indexed by DS:SI after the conversion, DS:SI will point to the least significant byte (digit).
Example: FFH Stored in AL register will be unpacked into 3 unpacked BCD digits 2,5, and 5 and will be stored at location 0:3000H, 3001H and 3002H respectively (assuming DS=0 and SI=3000H.)
If the number is less than 100 (64H), then the most significant digit is taken to be 0 for e.g. 63H (decimal=99) will be unpacked as 0,9 and 9.
Calling address: **FE00:018B**

9.2 MONITOR ROUNTINES INDEPENDENT OF OPERATING MODE

The following routines are useful for independent contorl of I/O devices connected to the trainer. These outine can thus be used regardless of the operating mode of ESA 86/88E

1. Name of routine: **SER_SEND_STRING**
   Function: Outputs a string of ASCII characters stored in memory to the console. The string is indexed by ES:SI where is the segment value and SI is the offset address of the starting location of the string. The string is terminated by 00H
   Calling address: **FE00:01AF**

2. Name of routine: **LCD_SEND_STRING**
   Function: Outputs a string of ASCII Characters stored in memory to the LCD. The string is indexed by ES:SI where ES is the segment value and SI is the offset address of the starting location of the string. The string is terminated by 00H
   Calling address: **FE00:01CE**

3. Name of routine: **CLR_LCD**
   Function: Clears the LCD interfaced with the trainer in either mode of operation.
   Calling address: **FE00:01ED**

4. Name of routine: **PRINT_STRING**
   Function: Prints a string of ASCII characters in memory when a parallel printer is interfaced with the trainer. The string is indexed by DS:SI where DS is the segment value and SI is the offset address of the starting location of the string. The string is terminated by 00H. Note that this routine is used for direct output to the printer regardless of whether the printer driver is enabled by DIP Switch.
   Calling address: **FE00:01F3**

CHAPTER 10

PROGRAMMING EXAMPLES

This chapter describes some programming examples that can be executed on the ESA 86/88E trainer. These examples range from fairly simple ones designed to illustrate the use of various instructions to some comprehensive examples designed to illustrate the use of monitor routines and demonstration examples for onboard peripherals. It is strongly urged that the user read this chapter carefully to be able to use ESA 86/88E efficiently.

The examples are presented in a format that makes it convenient for the user to enter programs using the monitor resident ESA 86/88E Symbolic One-line Assmebler. The HEX equivalents of the instructions are also given in the examples as they appear during disassembly. Thus the user may use Substitute Memory commands and enter the HEX codes at the memory locations directly, if so desired.

NOTE: User of RAM starts from 0:2000H and program entry or execution should not begin from an address within this area.

General instructions

1. Enter thr programs in the trainer memory at the locations shown along with the program using ESA 86/88E Symbolic One-Line assembler.
2. Some programs may require look-up tables. The corresponding data may be entered at the appropriate locations by using 'DB', 'DW' or 'ASC' directives, or by using the Substitute Memory Commands.
3. Using the GO command execute the program at its starting location. In most cases, the control returns to the monitor after program execution (because of the Breakpoint instruction INT 3). It is important to properly terminate any user program, failing which the program data may be lost.
4. After observing the results of these programs, it is urged that the user try different variations of these programs to get familiar with the ESA 86/88E system.
5. The actual disassembly of the programs will not contain some of the labels used; instead their reference locations will be displayed. However, the user may use these labels while assembly.

10.1 FAMILIARIZATION EXAMPLES

These examples are designed to familiarize the user with the operations of ESA86/88E system.

Example1 : This program computes the average of given word values stored in memory. The computed average is also stored at a given memory location. Note that this program does not check for overflow while forming the sum of the data values.

ADDRESS	OBJECT CODE	LABELS	MNEMONICS		COMMENTS
0000:2000	B8 00 00		MOV	AX, 0000	;Initialize
0000:2003	BE 20 20		MOV	SI, 2020	
0000:2006	8E C8		MOV	CS, AX	;segment registers
0000:2008	B9 05 00		MOV	CX, 05	;Load Count and
0000:200B	03 04	BAK:	ADD	AX, {SI}	;add the words
0000:200D	46		INC	SI	;sequentially
0000:200E	46		INC	SI	
0000:200F	E2 FA		LOOP	BAK	
0000:2011	B9 05 00		MOV	CX,05	
0000:2014	F7 F1		DIV	CX	;Divide Sum by
0000:2016	BE 30 20		MOV	SI,2030	;count
0000:2019	89 04		MOV	[SI],AX	;Store Computed
0000:201B	CC		INT	03	;average in memory
0000:201C			ORG	2020	;Data Words stored
0000:2020			DW	1000	;at 0:2020H
0000:2022			DW	2000	
0000:2024			DW	3000	
0000:2026			DW	4000	
0000:2028			DW	5000	

- This program will compute the average of 5 data words entered at locations 0:2020H onwards. The results is stored at memoery location 0:2030H
- Examine the contents of the word location RESULT (2030H). For the entries shown in the program, the result will be 3000H.

Example 2 : The following program exchanges two blocks of data stored in memory using the powerful string instructions of 8086/8088. This program exchanges 1FH bytes from 0:3000H and 0:3200H onwards. The user can use this program with necessary modifications to exchange data blocks of desired size between desired locations.

ADDRESS	OBJECT CODE	LABELS	MNEMONICS		COMMENTS
0000:2000	BB 00 30		MOV	BX,3000	;Store index
0000:2003	BA 00 32		MOV	DX,3200	;references in
0000:2006	B8 00 35		MOV	AX,3500	;registers
0000:2009	8B F3		MOV	SI, BX	;set up 1st block
0000:200B	8B F8		MOV	DI, AX	;as source & a temp
0000:200D	B9 20 00		MOV	CX, 20	;block as destination
0000:2010	FC		CLD		
0000:2011	F3		REP		
0000:2012	A4		MOVSB		

```
0000:2013   8B   F2        MOV   SI, DX      ;Set up 2nd block as
0000:2015   8B   FB        MOV   DI, BX ; source & 1st block as
0000:2017   B9   20   00   MOV   CX 20   ;Destination address
0000:201A   FC             CLD
0000:201B   F3             REP
0000:201C   A4             MOVSB
0000:201D   8B   F0        MOV   SI, AX   ;Set up temp, block as
0000:201F   8B   FA        MOV   DI, DX   ; source & 2st block as
0000:2021   B9   20   00   MOV   CX, 20   ;Destination address
0000:2024   FC             CLD
0000:2025   F3             REP
0000:2026   A4             MOVSB
0000:2027   CC             INT       30
```

- Enter the above program beginning at location 0:2000H
- Enter the desired data in locations 0:3000 to 0:3000 to 0:301FH and 0:3200H to 0:321FH
- Execute the program using GO command. Now the data block at 0:3000 would be moved to 0:3200h and the data block at 0:3200H to 0:3000H. This can be verified using display memory commands.

Example 3 : The following program coverts a hexadecimal byte value to its ASCII notation. The example illustates tge use of the powerful translate (XLAT) and rotate instructions. The program asumes that the hex value is in AL register. The resulting ASCII representation is left in the AX register. Enter the program at 0:2000H and enter the required HEX value to be converted in AL register.

```
ADDRESS        OBJECT CODE   LABELS   MNEMONICS       COMMENTS
0000:2000   BF 12   20          MOV    DI,2012     ;Get address  of ASCII
0000:2003   8D 1D               LEA    BX,[DI]     ;look-up table
0000:2005   32 E4               XOR    AH,AH       ;Clear upper byte
0000:2007   B1 04               MOV    CL,04
0000:2009   D3 C8               ROR    AX,CL       ;Lower nibble in AH
0000:200B   D2 CC               ROR    AH,CL       ;Upper nibble in AL
0000:200D   D7                  XLAT               ;ASCII code of upper
0000:200E   86 E0               XCHG   AH,AL       ;nibble in AH
0000:2010   D7                  XLAT               ;ASCII code of lower
0000:2011   CC                  INT    03          ;nibble in AL
0000:2012   30 31               DB     30, 31      ;ASCII code look-up
0000:2014   32 33               DB     32, 33      ;table
0000:2016   34 35               DB     34, 35
0000:2018   36 37               DB     36, 37
0000:201A   38 39               DB     38, 39
0000:201C   41 42               DB     41, 42
0000:201E   43 44               DB     43, 44
0000:2020   45 46               DB     45, 46
```

Example 4: This is a program to find the factorial of a given number. The program uses the Arithmetic instructions of the 8086/8088 CPU instruction set. The program does not take into account the carry generated by the multiplication and hence the factorial of smaller numbers (up to 8) can be properly calculated.

```
ADDRESS      OBJECT CODE   LABELS   MNEMONICS  COMMENTS
0000:2000    B8   00   00           MOV AX,0000 ;Initialize Segment
```

```
0000:2003   8E C8              MOV   CS,AX        ;registers
0000:2005   8E D8              MOV   DS,AX
0000:2007   BE 00 30           MOV   SI,300
0000:200A   8E 04              MOV   AL,[SI]      ;Load value from
                                                  ;memory
0000:200C   8A 1C        NXT:  MOV   BL,[SI]     ;save value and sub.
0000:200E   FE CB              DEC   BL           ;for repetitive
0000:2010   74 06              JZ    OVR          ;multiplication
0000:2012   88 1C              MOV   [SI],BL
0000:2014   F7 24              MULW  [SI]
0000:2016   EB F4              JMP   NXT
0000:2018   BF 00 31     OVR:  MOV   DI,3100      ;Store Computed
                                                  ;value
0000:201B   89 05              MOV   [DI],AX      ; in memory
0000:201D   CC                 INT   03
```

- After enetering the above program in memory,store the number whose factorial is to be computed at location 0:3000H.
- Then execute the program from its starting (0:2000H) and check for the computed value at location 0:3100H. The factorial will be in HEX format
- The user is urged to modify the program taking into account the carry generated during multiplication of larger numbers and verify the results

Example 5 : This program finds the largest value in a string of data bytes. The string of bytes is pointed by SI register, BH register holds the count value and the result is store in AL register.

```
ADDRESS     OBJECT CODE   LABELS  MNEMONICS  COMMENTS

0000:2000   B8 00 00        MOV   AX,0000     ;Initialize segment
0000:2003   8E C8                 MOV   CS,AX        ;registers
0000:2005   BE 00 21              MOV   SI, 2100     ;Initialize Pointer
0000:2008   B7 0A                 MOV   BH, 0A       ;Load count value
0000:200A   8A 04                 MOV   AL, [SI]     ;Load byte
0000:200C   46              NXT:  INC   SI
0000:200D   FE CF                 DEC   BH       ; Check 10 bytes in all
0000:200F   74 0A                 JZ    201B
0000:2011   8A 1C                 MOV   BL,[SI]      ; Compare current
                                                     ; value
0000:2013   3A C3                 CMP   AL, BL       ; with next vlaue
0000:2015   77 F5                 JNBE  NXT          ;If next value is
0000:2017   8A C3                 MOV   AL,BL    ;larger, load it in AL
0000:2019   EB F1                 JMP   NXT      ; Repeat the process
0000:201B   CC              OVR:  INT   03       ; Return to monitor
```

- Enter the above program in memory, and store a string of 10 bytes starting from 0:2100H location
- Execute the program from its starting address (0:2000H) and check the results. The user may then modify the program taking into account a string of word values

10.2 ILLUSTRATION OF ESA 86/88E MONITOR ROUTINES

These examples make use of some of ESA 86/88E monitor routines listed in Chapter 9. It is recommended that the user go through these routines to get familiar with the usage of these routines.

Example 1: Program to display a message "HELLO WORLD !" This program stores the HEX equivalent of the ASCII characters in AL register from memory indexed by SI register.

The program makes use of OUT-CHAR and GET_CHAR routines. If the program is excuted in serial mode the message is displayed on the console. In case the program is executed in stand-alone mode, then the output can be observed on the LCD. Then the program waits for the user to enter any key after which it returns to the monitor

ADDRESS	OBJECT CODE	LABELS	MNEMONICS	COMMENTS
0000:2000	BE 00 25		MOV SI ,2500	; Set up memory
0000:2003	B1 0E		MOV CL, 0E	; pointer and count
0000:2005	8A 04		MOV AL, [SI]	
0000:2007	9A 00 00	00 FE	CALLS 0FE00:0000	;Call OUT_CHAR
0000:200C	46		INC SI	;routine
0000:200D	E2 F6		LOOP 2005	;Repeat display
0000:200F	9A A9 00	00 FE	CALLS 0FE00:00A9	;Call GET_CHAR
0000:2014	CC		INT 03	; Return to monitor

The user should fill the locations from 0:2500H to 0:250DH with the following Hexadecimal data bytes. These bytes are the HEX equivalents of the ASCII characters contained in the output message.

0000:2500 0A, 0A, 48, 45, 4C, 4C, 4F, 20, 57, 4F, 52, 4C, 44, 21, 20.

Example 2 : This program demonstrates the use of GET_STRING and SEND_STRING routines. The program prompts the use to enter a message from the keyboard in either mode and output the same message on the console.

The output of the program will appear only on the console. Execute this program in stand-alone mode, and also connect the trainer to a CRT or PC using RS 232 C serial interface. If the communication package is running, the message entered via the PC keyboard interfaced to the trainer will be displayed on the console.

ADDRESS	OBJECT CODE	LABELS	MNEMONICS	COMMENTS
0000:4000	B8 00 00		MOV AX ,0000	; Initialize segment
0000:4003	8E C0		MOV ES, AX	; registers
0000:4005	B8 50 40		MOV AX,4050	;setup up mem pointer
0000:4008	9A 31 00	00 FE	CALLS 0FE00:0031	; for message ;prompt
0000:400D	9A 13 00	00 FE	INC SI	;Output message
0000:4012	9A 31 00	00 FE	LOOP 2005	;on LCD

```
0000:4017   9A 0E     01    00 FE   CALLS 0FE00:010E ;Call GET_STRING
0000:401C   8B C6                   MOV   AX,SI      ; routine and accept
0000:401E   9A 31     00    00 FE   CALLS 0FE00:0031 ; a message
0000:4023   9A AF     01    00 FE   CALLS  0FE00:01AF;Call
                                                     ;SER_SEND_STRING
0000:4028   EA 00     00    00 F0   JMPS  0F000:0000 ;routine to
                                                     ;output
                                                     ;mesage on the
                                                     ;console
0000:402D                           ORG    4050  ;Stored message string
0000:4050   45 4E     54 45 52 20   ASC     'ENTER A MESSAGE'
0000:4056   41 20     4D 45 53 53
0000:405C   41 47     45
0000:405F   00                      DB     00
```

Example 3: This program makes use of some conversion routine like ASCII_HEX and VALID_HEX in addition to other input/output routines The program waits for the user to enter a valid ASCII character in HEX. If the input is valid (31-39/41-46), the program output the equivalent character as a HEX value and repeats the sequence. If the user enters any other value the program displays the message 'INVALID INPUT' and returns control to ESA86/88E monitor. The program may be exeuted in serial or stand-alone mode of operation.

```
ADRESS      OBJECT      CODE    LABELS      MNEMONICS          COMMENTS
0000:2000   B8 00       00              MOV   AX,0000          ; Initialize segment
0000:2003   8E C0                       MOV   ES,AX          ;register
0000:2005   9A 31 00    00      FE      CALLSOFE00:0031      ;callout_crlf
                                                             ;to
                                                             ;display on new
                                                             ;line
0000:200A    B8 00  21                  MOV       AX,2100   ;display message
                                                             ;using
0000:200D    9A 13 00   00      FE      CALLS    0FE00:0013  ;OUT_STRING
                                                             ;routine
0000:2012   9A  E0 00   00      FE      CALLS    0FE00: 00E0 ; use get_word
                                                             ;routine
0000:2017   8A  E0                      MOV   AH,AL          ;to read user input
0000:2019   8A    D8                    MOV   BL,AL          ;check if user i/p
                                                             ; is a valid HEX
0000:201B   9A    23        0100  FE      CALLS 0FE0:0123    ; character
0000:2020   3C    00                    CMP   AL,00          ;in ASCII using
0000:2022   74    1B                    JE    INV            ;valid_hex routine
0000:2024   8A    C3                    MOV   AL,BL
0000:2026   3C    60                    CMP   AL,60
0000:2028   77    15                    JNBE  INV
0000:202A   B8    50  21                MOV AX,2150          ;display
                                                             ;massage
0000:202D    9A   13  00    00      FE      CALLS 0FE00:0013  ;if input is
                                                             ; valid
0000:2032   8A    E3                    MOV AH,BL            ;convert ASCII
                                                             ;value
0000:2034   9A    31  01    00      FE      CALLS 0FE00:0131  ; to hex using
                                                             ; hex_ASCII
```

```
ADRESS     OBJECT     CODE   LABELS    MNEMONICS          COMMENTS
                                                          ;routine
0000:2039  9A 52 00      00 FE          CALLS 0FE00:0052  ;Output binary
                                                          ;value
0000:203E  CC                           INT     3          ;Save regs. and
                                                          ; exit
0000:203F  B8 30 21              INV    MOV AX,2130        ; If user i/p is
                                                           ;Invalid
0000:2042  9A 13 00      00 FE          CALLS; 0FE00:0013 ; display
                                                          ; appropriate
0000:2047  CC                           JMP     2000      ; message and repeat
                                        CRG     2100       ; Display message
                                                          ; strings in memory

0000:2100  45     4E     54 45   52 20  ASC  'ENTER VALID HEX
                                         CHARACTER IN ASCII -'
0000:2106  56     41     4C 49   44 20
0000:210C  48     45     58 20   43 48
0000:2112  41     52     41 43   54 45
0000:2118  52     20     49 4E   20 41
0000:211E  53     43     49 49   20 2D
0000:2124  20     00                    DB      20,00
                                        ORG     2130
0000:2130  0A     0D                    DB      0A,OD
0000:2132  49     4E     56 41   4C 49  ASC     'INVALID INPUT'
0000:2138  44     20     49 4E   50 55
0000:213E  54
0000:213F  00                           DB      00
0000:2140                               ORG     2150
0000:2150  0A     0D                    DB      0A,0D
0000:2152  48     45     58 20   56 41  ASC     'HEX VALUE='
0000:2158  4C     55     45 20   3D
0000:215D  20     00                    DB      20,00
```

10.3 PROGRAMMING WITH ONBOARD HARDWARE

10.3.1 Use of KBINT key

This program section illustrates the use of KBINT key provided on the trainer. As explained in chapter 5, this key is connected to NMI or Type 2 interrupt of 8086/8088 CPU. The user has to provide the vectoring information for this interrupt . This program sets the vector locations for the type 2 interrupt and then in a loop. The code segment of the ISR is set at location 0:000AH as 0000H and the Instruction Pointer is set at location 0:0008H as 2100H.

```
ADDRESS   OBJECT CODE        LABELS  MNEMONICS      COMMENTS

0000:2000 B8 08    00                MOV    AX, 0008  ;Load ISR offset
                                                      ; value
0000:2003 8B F0                      MOV    SI, AX     ;at 0:0008H
0000:2005 B8 00    21                MOV    AX,2100
0000:2008 89 04                      MOV    [SI],AX
0000:200A B8 0A    00                MOV    AX,000A   ;Load ISR segment
```

```
0000:200D   8B   F0              MOV     SI, AX      ;value at 0:000AH
0000:200F   B8   00  00         MOV     AX, 0000
0000:2012   89   04              MOV     [SI],AX
0000:2014   8E   C0              MOV     ES, AX
0000:2016   B8   50  20          MOV     AX,2050     ; Set up memory
                                                     ;pointer
0000:2019   9A   31  00  00  FE  CALLS   OFE00:0031  ;Display memory
0000:201E   9A   13  00  00  FE  CALLS   OFE00:0013
0000:2023   EB   FE          LP: JMP     LP          ;Wait for
                                                     ;interrupt
```

After entering the above program, enter the following interrupt service routine at location 0:2100 H onwards. The message string at the end of the routine may be entered using Substitute Memory command also, using the HEX equivalents of the ASCII characters given.

```
ADDRESS     OBJECT CODE     LABELS  MNEMONICS       COMMENTS

0000:2100   B8  00   00             MOV  AX,0000 ;Initialize
                                                  ;segment
0000:2103   8E  C0                  MOV  ES,AX    ;registers
0000:2105   B8  67   20             MOV  AX,2067  ;Set up memory pointer
0000:2108   9A  31   00  00  FE     CALLS 0FE00:0031
0000:210D   9A  13   00  00  FE     CALLS 0F00:0013  ;Display;message
0000:2112   9A  A9   00  00  FE     CALLS 0FE00:00A9 ; Wait for user
0000:2117   EA  00   00  00  F0     JMPS  0F000:0000 ; Strobe and
                                                  ;end
                                    ORG   2050
0000:2050   57  41   49  54  49  4E ASC  'WAITING FOR'
0000:2056   47  20   46  4F  52
0000:205B   0A  0D                  DB    0A,0D
0000:205D   42  52   45  41  4B     ASC   'BREAK KEY'
0000:2062   20  4B   4B  45  59
0000:2066   00  0A                  DB    00 0A
0000:2068   42  52   45  41  4B  20 ASC  'BREAK KEY ACEPTED !'
0000:206E   4B  45   59  41  43  43
0000:2074   45  50   54  45  44  21
0000:207A   00                      DB    00
```

Now, execute the program from 0:2000H using the GO command. The Program will output a message **WAITING FOR BREAK KEY** and will wait for a user strobe. When the presses the KBINT key on the trainer, the message **BREAK KEY ACCEPTED!** is displayed. The program now waits for another user strobe before returning to command entry mode.

NOTE: The user can independently write vectorting information for Type 2 interrupt at location 0:0008H and 0:000AH using substitute memory commands. Pressing KBINT key anytime will then tranfer control to the program at the address specified by the content of these locations.

10.3.2 PROGRAMMABLE INTERVAL TIMER, 8253

ESA 86/88E provides the user with three independent timing channels via an onboard Programmable Interval Timer 8253. Of these, Timer 1 and 2 are fully available to the user. Refer Chapter 6 for the I/O address mapping of the timer. Clock, Gate and Output signals of these timers are brought out on a 8-pin header J9.

The following program module initializes TIMER 1 of 8253 A in Mode 3(Square Wave Generator). After entering and executing this program, user can observe a square waveform at TIMER1 of connector J9 on an oscilloscope with a time base of 5ms. Note that the user should give clock input to CLK1. The GATE 1 should be pulled high for MODE3 operation.

```
ADDRESS     OBJECT     CODE      LABELS     MNEMONICS     COMMENTS

0000:2000   B8 00      00        MOV        AX,0000       ;Initialize Segment
0000:2003   8E C8                MOV        CS,AX         ; registers
0000:2005   BA FF       FF       MOV        DX,0FFFF
0000:2008   B0 76                MOV        AL,76         ;Control Word for
0000:200A   EE                   OUT        DX,AL  ;TIMER 1 in MODE 3
0000:200B   BA FB       FF       MOV        DX,OFFFB      ;Load 16-bit Count
0000:200E   B0 10                MOV        AL,10         ;in timer 1 count
0000:2010   EE                   OUT        DX,AL         ;register
0000:2011   EE                   OUT        DX,AL
0000:2012   EB FE       NOW:     JMP        NOW
```

10.3.3 Onboard Programmable Peripheral Interface, 8255

ESA 86/88E provided the user with 48 programmable I/O lines two programmable peripheral interface, viz. 8255 ICs at U1 and U20. these lines are brought to 26-pin connectors J4 and J5 respectively. The user may connect any interface module compatible to these connectors and program the corresponding PPI to work with the interface. The addresses of control and data port are given in Chapter 6. Note that the 8255 at U20 and its corresponding connector is supplied only if the user opts of it

This is a demonstration program for Stepper Motor Interface assumed to be connected over connector J4 of the trainer (correspinding to 8255-PPI Low .) The interface can be obtained from ESA Pvt. Ltd., Bangalore as an option.

```
ADDRESS     OBJECT     CODE      LABELS  MNEMONICS     COMMENTS

0000:2000   BA E6      FF        MOV     DX,0FFE6      ;Initialize all 8255
0000:2003   BO 80                MOV     AL,80         ; Ports as output
0000:2005   EE                   OUT     DX,AL
0000:2006   BA E0      FF        OUT     DX,0FFE0
0000:2009   B0 88                MOV     AL,88         ;Output data to ports
0000:200B   EE                   OUT     DX,AL
0000:200C   E8 04      00        CALL    2013          ;Introduce delay
0000:200F   D0 C8                ROR     AL,1          ;Rotate data byte for
0000:2011   EB F8                JMP     200B          ;rotation of motor
0000:2013   B9 00      40        MOV     CX,4000       ;Delay subroutine
0000:2016   E2 FE          RPT:  LOOP    RPT
0000:2018   C3                   RET
```

Enter the program from 0:2000H onwards, execute it from this location and observe the behavior of the stepper motor. Change the data byte at 0:2010 from C8H to C0H and observe the results.

10.4 USE OF 8087 CO-PROCESSOR

ESA 86/88E provides direct support for an optional Numeric Data processor-8087. To utilize this feature, the user has to simple install 8087 IC in the socket provided (U21). No other hardware changes are required. The following two example illustrate the use of 8087.

NOTE: ESA 86/88E Symbolic One-line Assembler does not support NDP instructions. so these programs must be entered directly in machine code (hexadecimal values) using Substitute Memory Command.

Example 1: The following program assumes that two 32-bit integer data values, a and b are stored at locations 3000H and 3004H onwards respectively. It then computes a result C= square root of (a^2+b^2) and stores the result as a 32-bit integer starting at location 3008H. (code segment value is assumed as 0000)

ADDRESS	OBJECT	CODE	LABELS	MNEMONICS	COMMENTS
0000:2000	BB 00	30		MOV BX,3000	;Point to 'a'
0000:2003	DB 07			FLD (BX)	;Load 'a'
0000:2005	9B			FWAIT	
0000:2006	DA 0F			FMUL (BX)	;Compute a2
0000:2008	9B			FWAIT	
0000:2009	DD D9			FST ST(1)	;save a2 in ST (1)
0000:200B	9B			FWAIT	
0000:200C	81 D3 04	00		ADD BX,0004	;Point to 'b'
0000:2010	DB 07			FLD (BX)	;Load 'b'
0000:2012	9B			FWAIT	
0000:2013	DA OF			FMUL (BX)	;Computer b2
0000:2015	9B			FWAIT	
0000:2016	D8 C1			FADD ST (1)	;ST(0) = a2+b2
0000:2018	9B			FWAIT	
0000:2019	D9 FA			FSQRT	;ST (0) = Sqrt (a2+b2)
0000:201B	9B			FWAIT	
0000:201C	81 D3 04	00		ADD BX,0004	; Point to location
0000:2020	DB 17			FST (BX)	; for C and store the
0000:2022	9B			FWAIT	;result as 32-bit
0000:2023	CC			INT 3	;integer in C & exit

1. Load the above program into memory.
2. Set up data values a and b as follows.

   0000:3000: 03, 00, 00, 00
   0000:3004: 04, 00, 00, 00
3. Executes the program and observe the result. It should be as follows:

   0000:3008: 05, 00, 00, 00
4. The user is urged to try with different data values to observe the results.

Example 2: The following program calculates Sin (Z) where Z is in degrees and 0<Z< 90. Note that the values 0 and 90 are not allowed. They must be handled separately.

Sin (Z) is calculated using the Tangent function (FPTAN) of 8087 as follows:
Suppose tan (Z/2)=Y/X , then sin (Z) =2XY/(X^2+Y^2)

The FPTAN function of 8087 requires the argument, in radians on the top of the stack and returns the result as Y/X where X is the top of stack and Y is the next element. Further the argument must satisfy the condition 0 < argument <PI/4. Hence Z is restricted to satisfy 0<Z <90. This restriction can be eliminated at the cost of more computation.

If Z be unrestricted, then the FPREM function of 8087 can be used to reduce the argument to the required range and use the relation sin (PI+Z) = Sin (Z)

In the following program, the argrument Z is first divided by 2(preparatory to using the relation described above), then converted into radians and then the tangent is calculated. The resulting values X and Y is used to calculate sin (Z). The input to the program, Z (in degreess) must be set up in the register AL , as a Hex value . The output, in packed BCD form is available in the register AX. A decimal point is to assumed before the first BCD digit.

```
ADDRESS     OBJECT CODE LABELS   MNEMONICS              COMMENTS

0000:2000   B4 00                MOV   AH,00
0000:2002   D1 E8                SHR   AX,1             ;Z=Angle = Angle/2
0000:2004   A3 5A      20        MOV   Arg1,AX          ;Preparatory for
0000:2007   9B DF      06 5A 20  FLDI  Arg1,            ;loading into NDP
0000:200C   C7 06      5C 20 B4  00  MOV   Arg2, 00B4   ;(Arg2.)=180
0000:2012   9B DF      06 5C 20  FLDI  Arg2,            ; Insert a FWAIT and
                                                        ; Load Arg2.
0000:2017   9B DE      F9        FDIVRP                 ; wait and divide (ST)
                                                        ;=z/180
0000:201A   9B D9      EB        FLDPI                  ;(ST) =PI
0000:201D   9B DE      C9        MULRP                  ;(ST) =PI *Z/80 =Angle
                                                        ;in radians
0000:2020   9B D9      F2        FPTAN                  ; Conmpute tan as Y/X
0000:2023   9B D9      C0        FLD   ST(0)            ; copy x onto stack top
0000:2026   9B D9      C9        FMUL ST ,ST(1)         ; ST =X*X
0000:2029   9B D9      C2        FLD   ST (2)           ;ST = Y
0000:202C   9B D8      CB        FMUL  ST,ST(3)         ; Y*Y
0000:200F   9B DE      C1        FADDP ST(1),ST         ;Y*Y is popped off
                                                        ; ST =x*x+y*y
0000:2032   9B DE      F9        FDIVRP                 ;ST = x/(x*x+y*y)
0000:2035   9B DE      C9        FMULRP                 ; ST = x*x/cx*x+y*y)
0000:2038   9B D9      C0        FLD ST (0)             ; Copy value onto
                                                        ; stack
0000:203B   9B DE      C1        FADDP ST (1),ST        ; ST= 2*X*Y[X*X-
                                                        ; y*y)
0000:203E   C7 06      5E        20 10 27  MOV Arg 3, 2710
0000:2044   9B DF      06        5E 20     FLDI Arg 3
0000:2049   9B DE      C9        FMURP                  ; Result=
```

```
                                                    ;1000*SIN(Z)
0000:204C   9B D9   FC                  FRNDINT     ;Round to integer
0000:204F   9B DF   36 60 20            FSTP Arg4   ;and store as BCD
0000:2054   9B                          FWAIT       ;Get the result into
0000:2055   A1 60   20                  MOV AX, Arg4  ; Acumulator
0000:2058   CC                          INT 3       ; Return to monitor
```

Note: 16 locations from 0:205 AH to 206AH (ARGI, ARG2, ARG3 and ARG4) are used for storing the constants and results.

1. Load the above program into the memory.
2. Set up the input parameter in the register AL (for example, to calculate Sin 60,(AL,)= 3CH)
3. Execute the program and observe the contents of AX [with input as (AL)=3C, (AX) will now be 8660; so sin 60=0.8660]
4. Repeat the program with different input data and observe the output as shown below.
5. The user may try using the FPREM function of 8087, calculate sin Z for any value of Z

Input (AL)	Output(AX)	Calculated function
1E	5000	sin 30
2E	7193	sin 46
10	2756	sin 16
56	9976	sin 86

10.5 PROGRAMMING WITH INTERFACES

10.5.1 Parallel Printer Inteface

This program demostrates direct output to the printer using PRINT_STRING rountine. Using this program, parllel printer interfaced with the trainer can be acessed regrardless of the printer enable DIP switch SW5.The printer cable is connected over the 26-pin FRC connector J4. the necessary cable may be obtained from ESA Pvt. LTD, Bangalore as an option .

This program sends message **HELLO WORLD!** directly to the printer. This routine can be called from the user's program when the system is operating in either of the two modes. Refer Chapter 9 for a description of this routine and it's calling address.

```
ADDRESS     OBJECT CODE     LABELS   MNEMONICS          COMMENTS

0000:2500   B8 00 00                  MOV AX,0000       ;Initialize segment
0000:2503   8E D8                     MOV DS,AX         ; registers
0000:2505   BE 00 30                  MOV SI,300        ; Set up memory
                                                        ; Pointer to
message
0000:2508   9A F3   01  00    FE      CALLS 0FE00:01F3 ;Call
                                                        PRINT_STRING
0000:250D   CC                        INT 03            ; routine and
                                                        ; terminate.
```

Along with the above program, enter the following look up table for the message string from location 0: 3000H. Then, execute the program and observe the output string printed by the printer. 0000:3000 0A, 0D, 20, 20, 48, 45, 4C, 4F, 57, 4F, 52, 4C, 44, 21,00 This program makes use of the monitor routine to print the line of characters. Alternativaly the user may write programs to output to the printer by using the information given in Section 6.7

10.5.2 Liquid Crystal Display Interface

This example illustrates the initialization of the onboard LCD available with ESA 86/88E Trainer. The 8255 positioned at U2 control the LCD.The details of thle instruction and data word formats for LCD may be obtained from Crystalonics Displays User Manul.

This program displays the string **HELLO WORLD !** on the LCD continuously with a specific delay. The user can utilize this program as a model to develop his own projects using the LCD.This program can be exeuted in either mode of trainer operation. Enter the program from 0:2000H location onwards. Excute the program and observe the output on the LCD.

ADDRESS	OBJECT CODE	LABELS	MNEMONICS		COMMENTS
0000:2000			CMD	EQU 2034	
0000:2000			STR	EQU 2026	
0000:2000			DWR	EQU 204A	
0000:2000			MES	EQU 2008	
0000:2000	BO 80		MOV	AL,80	;Initialise all 8255
0000:2002	BA DE FF		MOV	DX,OFFDE	; port as output
0000:2005	EE		OUT	DX,AL	
0000:2006	EB 0D		JMP	2015	
0000:2008	48 45 4C 4C 4F 20		MES:	ASC	'HELLOWWORLD!'
0000:200E	57 4F 52 4C 44 21				;message string
0000:2014	00		DB	00	
0000:2015	B0 0C	START:	MOV	AL,0C	;Display on, no
0000:2017	E8 1A 00		CALL	CMD	; cursor or blinking ; character
0000:201A	B0 01		MOV	AL,01	;Clear Display
0000:200C	E8 15 00		CALL	CMD	
0000:200F	2E		CS		
0000:2020	8D 16 08 20		LEA	DX,@MES	;Set up pointer to
0000:2024	8B DA	MOV	BX,DX		; message string
0000:2026	8A 07	STR:	MOV	AL,[BX]	
0000:2028	3C 00		CMP	AL,[BX]	
0000:200A	74 E9		JE	START	; termination
0000:202C	E8 1B 00		CALL	DWR	; Call routine to
0000:202F	43		INC	BX	; write data
0000:2030	EB F4		JMP	STR	
0000:2032	EB E1		JMP	STRAT	; Repeat continuously
0000:2034	BA D8 FF	CMD	MOV	DX,0FFD8	; Write into Data
0000:2037	EE		OUT	DX,AL	; regrister
0000:2038	BA DC FF		MOV	DX,0FFDC	; enable Read/Write
0000:203B	B0 06		MOV	AL,06	; In Instruction.

```
0000:203D   EE              OUT   DX,AL             ;register
0000:203E   B0  08          MOV   AL,08
0000:2040   EE              OUT   DX,AL
0000:2041   B0  00          MOV   AL,00
0000:2043   EE              OUT   DX,AL
0000:2044   B9  00  80      MOV   CX,8000
0000:2047   E2  EE    DY1:  LOOP  DY1
0000:2049   C3  FE          RET

0000:204A   BA  D8  FF      DWR:  MOV   DX,0FFED8   ;Routine to write
0000:204D   EE              OUT   DX,AL             ;data
0000:204E   B0  F4          MOV   DX,0F4
0000:2050   BA  DC  FF      MOV   DX, 0FFDC
0000:2053   EE              OUT   DX,AL
0000:2054   B0  F2          MOV   AL,0F2            ; Return to start of
0000:2056   EE              OUT   DX,AL             ; LCD disply
0000:2057   B0  FA          MOV   AL,0FA
0000:2059   EE              OUT   DX,AL
0000:205A   B0  F2          MOV   AL,0F2
0000:205C   EE              OUT   DX,AL
0000:205D   B9  00  80      MOV   CX,8000           ;Delay between
0000:2060   E2  FE    DY2:  LOOP  DY2               ;Repeated displys
0000:2062   C3              RET
```

This chapter covered a variety of programming example that were developed using ESA 86/88E Symbolic One-line assembler. The user in urged to try different variations of the program modules given here, so that the capabilities of the programming enviroment provided are fully appreciated. We welcome any recommendations or suggestions for improvement.